第一阶段面试题

1.接口和抽象类的异同点

相同：

- 都不能创建对象
- 都可以定义抽象方法，并且一定要在子类中重写

不同：

- 关键字不同abstractinterface
- 抽象方法中既可以有抽象的方法也可以有普通的方法
- 接口中中所有的方法都是抽象方法
- 抽象类的方法可以任意权限，接口中方法只能是public
- 抽象类只能单继承，接口可以多实现

2.重载(overload)和重写(overwrite)区别

重写的规则

子类和父类,子类重写了父类的方法

- 方法名、参数列表必须和父类完全一致
- 返回值类型要么相同，要么子类方法的返回值类型是父类方法返回值类型的子类!
- 访问修饰符要么相同，要么子类访问修饰符范围大于父类!
- 方法中抛出的异常，要么相同。要么子类方法抛出的异常比父类被重写方法抛出的异常更小或相同!

重载的规则(两同一不同)

1）同一个类中
2）方法名相同
3）参数列表不同（个数，列表，类型）
4）和返回值无关

3.StringBufferStringBuilderString区别

String字符串常量不可变使用字符串拼接时会开辟新空间

StringBuffer字符串变量可变线程安全字符串拼接直接在字符串后追加

StringBuilder字符串变量可变非线程安全字符串拼接直接在字符串后追加

1.StringBuilder执行效率高StringBuffer，高于String.

2.String是一个常量，是不可变的，所以对于每一次+=赋值都会创建一个新的对象，　　StringBuffer和StringBuilder都是可变的，当进行字符串拼接时采用append方　　法，在原来的基础上进行追加，所以性能比String要高，

StringBuffer是线程安全的而StringBuilder是线程非安全的，所以StringBuilder的效率高于StringBuffer.

3.对于大数据量的字符串的拼接，采用StringBuffer,StringBuilder.

4.冒泡排序

工具类版本：

5.选择排序

6.单例设计模式

单例就是该类只能返回一个实例。

单例所具备的特点：

1.私有化的构造函数

2.私有的静态的全局变量

3.公有的静态的方法

懒汉式：

线程安全懒汉式：

饿汉式：

7.常见的异常类型

NullPointerException空指针异常

ClassCastException类型强制转换异常

IllegalArgumentException传递非法参数异常

ArithmeticException算数运算异常

IndexOutOfBoundsException下标越界异常

NumberFormatException数字格式异常

ClassNotFindException加载请求异常

8.Throws和Throw的区别

- 位置：Throw方法内部，Throws方法名之后
- 作用：Throw抛出异常，Throws声明异常
- 个数：Throw一次抛出一个异常，Throws可以声明多个异常

9.Final，Finally，finalize关键字的作用

final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

finally是异常处理语句结构的一部分，表示总是执行。不管是否有异常总是被执行，除非虚拟机停止才不执行。
System.exit(1);

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法可以实现资源回收，释放资源，例如关闭文件等。JVM不保证此方法总被调用.

10.Final关键字的作用

- 修饰类：（最终类）不能被子类继承
- 修饰方法：（最终方法）不能被子类重写
- 修饰变量：（常量）一旦声明之后，不能再次修改其中的值

11.Hashtable与HashMap的区别

Hashmap 允许空（null）键值（key），非线程安全，效率高。在多个线程访问时必须提供外同步。

HashMap是Java1.2引进的Mapinterface的一个实现。

Hashtable不允许key为null值，线程安全，但是效率低。

HashMap把Hashtable的contains方法去掉了,改成containsvalue和containsKey。

Hashtable和HashMap采用的hash/rehash算法大致一样，所以性能不会有很大的差异。

12.线程和进程的区别

线程是指在程序执行过程中，能够执行程序代码的一个基本执行单位，每个程序至少都有一个线程，也就是程序本身。
Java中的线程有四种状态：运行、就绪、挂起、结束。
进程是程序的基本执行实体，是线线程的容器。
进程具有的特征：
动态性：进程是程序的一次执行过程，是临时的，有生命期的，是动态产生，动态消亡的；
并发性：任何进程都可以同其他进程一起并发执行；
独立性：进程是系统进行资源分配和调度的一个独立单位；
结构性：进程由程序、数据和进程控制块三部分组成。
线程和进程的区别：
1、隶属关系：线程是进程中的一部分，一个进程中可以包含若干个线程。进程可以申请和拥有系统资源，是一个动态的概念，是一个活动的实体。
2、地址空间和其它资源：进程间相互独立，同一进程的各线程间共享。某进程内的线程在其它进程不可见。在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位。
3、进程间通信：线程间可以直接读写进程数据段（如全局变量）来进行通信。需要进程同步和互斥手段的辅助，以保证数据的一致性。
4、调度和切换：线程上下文切换比进程上下文切换要快得多。

13.实现多线程程序的2种方式

多线程有两种实现方式：
一、继承Thread类。
二、实现Runnable接口。
同步的实现方法有两种：
Wait()使一个线程处于等待状态，并且释放所持有的对象的lock。
notify()：唤醒一个处于等待状态的线程，在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，不是按优先级。
用synchronized关键字修饰同步方法反对使用stop()，是因为它不安全。
suspend()方法容易发生死锁。调用suspend()的时候，目标线程会停下来，但仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被“挂起”的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。所以不应该使用suspend()，而应在自己的Thread类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用wait()命其进入等待状态。若标志指出线程应当恢复，则用一个notify()重新启动线程。

14.List,Set,Collection,Collections

1.List和Set都是接口，他们都继承于接口Collection,List是一个有序的可重复的集合，而Set是无序的不可重复的集合。Collection是集合的顶层接口，Collections是一个封装了众多关于集合操作的静态方法的工具类,因为构造方法是私有的，所以不能实例化。

2.List接口实现类有ArrayList,LinkedList,Vector。ArrayList和Vector是基于数组实现的,所以查询的时候速度快，而在进行增加和删除的时候速度较慢LinkedList是基于链式存储结构，所以在进行查询的时候速度较慢但在进行增加和删除的时候速度较快。又因为Vector是线程安全的，所以他和ArrayList相比而言，查询效率要低。

15.sleep()和wait()有什么区别

sleep是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用sleep不会释放对象锁。wait是Object类的方法，对此对象调用wait方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出notify方法（或notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态
相同点在于都会造成线程阻塞.

16.error和exception有什么区别?

error表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。
exception表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。

17.heap（堆）和stack（栈）有什么区别

java的内存分为两类，一类是栈内存，一类是堆内存。栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。
堆是与栈作用不同的内存，一般用于存放不放在当前方法栈中的那些数据，例如，使用new创建的对象都放在堆里，所以，它不会随方法的结束而消失。方法中的局部变量使用final修饰后，放在堆中，而不是栈中。

18.GC是什么?为什么要有GC?

GC是垃圾收集的意思（GabageCollection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。

19.内存泄漏和内存溢出

内存泄露(memoryleak)，是指应用程序在申请内存后，无法释放已经申请的内存空间.一次内存泄露危害可以忽略，但如果任其发展最终会导致内存溢出(outofmemory).如读取文件后流要进行及时的关闭以及对数据库连接的释放。
内存溢出（outofmemory）是指应用程序在申请内存时，没有足够的内存空间供其使用。如我们在项目中对于大批量数据的导入，采用分段批量提交的方式。

20.运行时异常和checked异常的区别?

java异常是程序运行过程中出现的错误。
运行时异常:都是RuntimeException类及其子类异常。
IndexOutOfBoundsException索引越界异常
ArithmeticException：数学计算异常
NullPointerException：空指针异常
ArrayOutOfBoundsException：数组索引越界异常
ClassNotFoundException：类文件未找到异常
ClassCastException：造型异常（类型转换异常）

这些异常是不检查异常（UncheckedException），程序中可以选择捕获处理，也可以不处理。这些异常一般是由程序逻辑错误引起的。

checked异常，又叫做非运行时异常:是RuntimeException以外的异常，类型上都属于Exception类及其子类。从程序语法角度讲是必须进行处理的异常，如果不处理，程序就不能编译通过。如：

```
    IOException、文件读写异常
        FileNotFoundException：文件未找到异常
        EOFException：读写文件尾异常
        MalformedURLException：URL格式错误异常
        SocketException：Socket异常
        SQLException：SQL数据库异常
```

21.四个访问修饰符合访问级别?

四个访问修饰符:private，default,protected，public

22.逻辑运算符:&和&&的区别?

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为true时，整个运算结果才为true，否则，只要有一方为false，则结果为false。
&&还具有短路的功能，即如果第一个表达式为false，则不再计算第二个表达式。
&还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作

23.Java中如何实现序列化，有什么意义

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决对象流读写操作时可能引发的问题（如果不进行序列化可能会存在数据乱序的问题）。

要实现序列化，需要让一个类实现Serializable接口，该接口是一个标识性接口，标注该类对象是可被序列化的，然后使用一个输出流来构造一个对象输出流并通过writeObject(Object)方法就可以将实现对象写出（即保存其状态）；如果需要反序列化则可以用一个输入流建立对象输入流，然后通过readObject方法从流中读取对象。序列化除了能够实现对象的持久化之外，还能够用于对象的深度克隆。

24.阐述JDBC操作数据库的步骤

```
- 加载驱动。
Class.forName("com.mysql.jdbc.Driver");
- 创建连接。
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/数据库名
称","root","123456");
- 创建语句。
PreparedStatement ps=con.prepareStatement("select*from emp where sal between?and?");
ps.setInt(1,1000);
ps.setInt(2,3000);
- 执行语句。
ResultSetrs=ps.executeQuery();
- 处理结果。
while(rs.next()){
System.out.println(rs.getInt("empno")+"-"+rs.getString("ename"));
}
-关闭资源。
finally{
if(con!=null){
try{
```

```
con.close();
}catch(SQLExceptione){
e.printStackTrace();
}
}
}
```

提示：关闭外部资源的顺序应该和打开的顺序相反，也就是说先关闭ResultSet、再关闭Statement、在关闭Connection。上面的代码只关闭了Connection（连接），虽然通常情况下在关闭连接时，连接上创建的语句和打开的游标也会关闭，但不能保证总是如此，因此应该按照刚才说的顺序分别关闭。此外，第一步加载驱动在JDBC4.0中是可以省略的（自动从类路径中加载驱动）

25.Statement和PreparedStatement有什么区别？哪个性能更好?

与Statement相比，

- PreparedStatement接口代表预编译的语句，它主要的优势在于可以减少SQL的编译错误并增加SQL的安全性（减少SQL注射攻击的可能性）；
- PreparedStatement中的SQL语句是可以带参数的，避免了用字符串连接拼接SQL语句的麻烦和不安全；
- 当批量处理SQL或频繁执行相同的查询时，PreparedStatement有明显的性能上的优势，由于数据库可以将编译优化后的SQL语句缓存起来，下次执行相同结构的语句时就会很快（不用再次编译和生成执行计划）。

26.二分查找法

1.二分查找又称折半查找，它是一种效率较高的查找方法。
2.二分查找要求(前提)：（1）必须采用顺序存储结构（2）.必须按关键字大小有序排列
3.原理：将数组分为三部分，依次是中值（所谓的中值就是数组中间位置的那个值）前，中值，中值后；将要查找的值和数组的中值进行比较，若小于中值则在中值前面找，若大于中值则在中值后面找，等于中值时直接返回。然后依次是一个递归过程，将前半部分或者后半部分继续分解为三部分。
4.代码实现

```
public static void main(String[]args){
                int[]array={1,4,7,9,10,44,78,101,203,500};
                int index=binarySearch(array,78);
                System.out.println(index);
        }

        /*
        *循环实现二分查找算法arr已排好序的数组x需要查找的数-1无法查到数据
        */
        public static int binarySearch(int[]arr,intx){
                intlow=0;
                inthigh=arr.length-1;
                while(low<=high){
                        intmiddle=(low+high)/2;
                        if(x==arr[middle]){
                                returnmiddle;
                        }elseif(x<arr[middle]){
                                high=middle-1;
                        }else{
                                low=middle+1;
                        }
                }
                return-1;
        }
```

27.提示输入一个数，求阶乘

```
public static void main(String[]args){
                System.out.print("请输入一个正整数");
                Scanner sc=new Scanner(System.in);
                int n=sc.nextInt();
                if(n<1){
                        System.out.println("无效数据!");
                        return;
                }
                System.out.print(n+"!=");
                int result=1;
                for(inti=1;i<=n;i++){
                        result=result*i;
                        if(i==n)
                                System.out.print(i+"=");
                        else
                                System.out.print(i+"x");
                }
                System.out.print(result);
        }
```

28.菲波那切数列

斐波纳契数列，又称黄金分割数列,数字上的体现是1,1,2,3,5,8,13……（实际上是前两个数之和等于第三个数）

29.Java中基本数据类型

```
数据类型  大小包装类
boolean(布尔型)  1位Boolean(布尔型)
byte(字节)       1(8位)Byte(字节)
char(字符型)     2(16位)Character(字符型)
shot(短整型)     2(16位)Shot(短整型)
int(整型)        4(32位)Integer(整型)
long(长整型)     8(32位)Long(整型)
float(浮点型)    4(32位)Float(浮点型)
double(双精度)   8(64位)Double(双精度)
```

30.遍历D盘下面所有的文件

```
public void testprint(Filefile){
                String name=file.getName();
                boolean b=file.isDirectory();//判断是否为文件夹
                if(b){
                        //是文件夹
                        File files[]=file.listFiles();
                        for(File f:files){
                                testprint(f);
                                //System.out.println(name);//打印文件夹名字
                        }
                }else{
                        System.out.println(name);
                }
```

```
        }

        public static void main(String[]args){
                File file=new File("d:\\");
                testfile fi=new testfile();
                fi.testprint(file);

        }
```

31.ArrayList和LinkedList有什么区别

1)因为Array是基于索引(index)的数据结构，它使用索引在数组中搜索和读取数据是很快的。Array获取数据的时间复杂度是O(1),但是要删除数据却是开销很大的，因为这需要重排数组中的所有数据。
2)相对于ArrayList，LinkedList插入是更快的。因为LinkedList不像ArrayList一样，不需要改变数组的大小，也不需要在数组装满的时候要将所有的数据重新装入一个新的数组，这是ArrayList最坏的一种情况，时间复杂度是O(n)，而LinkedList中插入或删除的时间复杂度仅为O(1)。ArrayList在插入数据时还需要更新索引（除了插入数组的尾部）。
3)类似于插入数据，删除数据时，LinkedList也优于ArrayList。
4)LinkedList需要更多的内存，因为ArrayList的每个索引的位置是实际的数据，而LinkedList中的每个节点中存储的是实际的数据和前后节点的位置。

32.面向对象的特征有哪些方面

1)抽象:
抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。
抽象包括两个方面，一是过程抽象，二是数据抽象。
2)继承:
继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。
对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。
3)封装:
封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。
面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。
4)多态:
多态是指允许不同类的对象对同一消息作出响应。
多态包括参数化多态性和包含多态性。
多态语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

33.String是最基本的数据类型吗?

基本数据类型包括byte、int、char、long、float、double、boolean和short。
java.lang.String类是final类型的，因此不可以继承这个类、不能修改这个类。为了提高效率节省空间，我们应该用StringBuffer类

34.有一个字符串，其中包含中文字符、英文字符和数字字符，请统计和打印出各个字符的个数

```
publicstaticvoidmain(String[]args){
                Stringstr="中国aadf的111萨bbb菲的zz萨菲";
                intengishCount=0;
                intchineseCount=0;
                intdigitCount=0;
```

```
                for(inti=0;i<str.length();i++){
                        charch=str.charAt(i);
                        if(ch>='0'&&ch<='9'){
                                digitCount++;
                        }elseif((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')){
                                engishCount++;
                        }else{
                                chineseCount++;
                        }
                }
                System.out.println("英文字符:"+engishCount);
                System.out.println("中文字符:"+chineseCount);
                System.out.println("数字字符:"+digitCount);
        }
```

35写一个程序将D盘下面的一张图片拷贝到E盘

```
/*
        *从d盘将文件拷贝到e盘
        */
        public static void main(String[]args){
                File f1=newFile("d:/J1610B面试题库0220.pdf");
                File f2=newFile("e:/J1610B面试题库0220.pdf");
                try{
                        forJava(f1,f2);
                        System.out.println("拷贝成功~");
                }catch(Exceptione){
                        e.printStackTrace();
                }
        }

        public static long forJava(Filef1,Filef2)throws Exception{
                long time=newDate().getTime();
                int length=2097152;
                FileInputStream in=new FileInputStream(f1);
                FileOutputStream out=new FileOutputStream(f2);
                byte[]buffer=new byte[length];
                while(true){
                        in tins=in.read(buffer);
                        if(ins==-1){
                                in.close();
                                out.flush();
                                out.close();
                                returnnewDate().getTime()-time;
                        }else{
                                out.write(buffer,0,ins);
                        }
                }
        }
```

36静态类型有什么特点?

(1)静态的属性：随着类的加载而加载，该属性不在属于某个对象，属于整个类
(2)静态的方法：直接用类名调用，静态方法里不能访问非静态成员变量
(3)静态类：不能直接创建对象，不可被继承

37.说一下多态的表现形式?

(1)重载,重写,重载Overload表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同
(2)重写Override表示子类中的方法可以与父类中的某个方法的名称和参数完全相同，通过子类创建的实例对象调用这个方法时，将调用子类中的定义方法，这相当于把父类中定义的那个完全相同的方法给覆盖了，这也是面向对象编程的多态性的一种表现,只能比父类抛出更少的异常，或者是抛出父类抛出的异常的子异常,子类方法的访问权限只能比父类的更大，不能更小。如果父类的方法是private类型，那么，子类则不存在覆盖的限制，相当于子类中增加了一个全新的方法

38.线程通常有五种状态

- 创建状态。在生成线程对象，并没有调用该对象的start方法，这是线程处于创建状态
- 就绪状态。当调用了线程对象的start方法之后，该线程就进入了就绪状态，但是此时线程调度程序还没有把该线程设置为当前线程，此时处于就绪状态。在线程运行之后，从等待或者睡眠中回来之后，也会处于就绪状态。
- 运行状态。线程调度程序将处于就绪状态的线程设置为当前线程，此时线程就进入了运行状态，开始运行run函数当中的代码。
- 阻塞状态。线程正在运行的时候，被暂停，通常是为了等待某个时间的发生(比如说某项资源就绪)之后再继续运行。sleep,suspend，wait等方法都可以导致线程阻塞。
- 死亡状态。如果一个线程的run方法执行结束或者调用stop方法后，该线程就会死亡。对于已经死亡的线程，无法再使用start方法令其进入就绪

39.事务的四大特性：

40.Java中IO体系：

41.字节流和字符流：

- Stream结尾都是字节流，reader和writer结尾都是字符流
- 两者的区别就是读写的时候一个是按字节读写，一个是按字符，实际使用通常差不多。
- 在读写文件需要对内容按行处理，比如比较特定字符，处理某一行数据的时候一般会选择字符流。
- 只是读写文件，和文件内容无关的，一般选择字节流。

42.接口是否可继承接口?抽象类是否可实现(implements)接口?抽象类是否可继承实体类(concreteclass)?

接口可以继承接口
- 抽象类可以实现接口
- 抽象类可以继承实体类

43.Class.forName的作用?为什么要用?

按参数中指定的字符串形式的类名去搜索并加载相应的类，如果该类字节码已经被加载过，则返回代表该字节码的Class实例对象，否则，按类加载器的委托机制去搜索和加载该类，如果所有的类加载器都无法加载到该类，则抛出ClassNotFoundException。加载完这个Class字节码后，接着就可以使用Class字节码的newInstance方法去创建该类的实例对象了。

有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时才能确定，这时候就需要使用Class.forName去动态加载该类，这个类名通常是在配置文件中配置的，例如，Spring的ioc中每次依赖注入的具体类就是这样配置的，jdbc的驱动类名通常也是通过配置文件来配置的，以便在产品交付使用后不用修改源程序就可以更换驱动类名。

44.异常的体系结构：

45.构造函数的特点和作用

- 方法名和类名相同
- 没有返回值，连void也没有
- 初始化对象

-

46.Java标识符的命名规则：

- 只能有字母，数字，下划线和$组成
- 数字在不能开头
- 不能和关键字重名
- 见名知义

-

47.Java关键字：

48.构造函数的特点和作用

- 方法名和类名相同
- 没有返回值，连void也没有
- 初始化对象

-
  -

49.synchronized关键字的用法?

synchronized关键字可以将对象或者方法标记为同步，以实现对对象和方法的互斥访问，可以用synchronized(对象){…}定义同步代码块，或者在声明方法时将synchronized作为方法的修饰符。

- 同步代码块
- 同步方法

50.启动一个线程是调用run()还是start()方法?

启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行，这并不意味着线程就会立即运行。
run()方法是线程启动后要进行回调（callback）的方法。

51.Swtich是否能作用在byte上，是否能作用在long上，是否能作用在String上?

在Java5以前，switch(expr)中，expr只能是byte、short、char、int。从Java5开始，Java中引入了枚举类型，expr也可以是enum类型，从Java7开始，expr还可以是字符串（String），但是长整型（long）在目前所有的版本中都是不可以的。

52.数组有没有length()方法？String有没有length()方法?

数组没有length()方法，有length的属性。String有length()方法。**JavaScript中，获得字符串的长度是通过length属性得到的，这一点容易和Java混淆

53.构造器（constructor）是否可被重写（override）？

构造器不能被继承，因此不能被重写，但可以被重载。

54.阐述静态变量和实例变量的区别

静态变量是被static修饰符修饰的变量，也称为类变量，它属于类，不属于类的任何一个对象，一个类不管创建多少个对象，静态变量在内存中有且仅有一个拷贝；
实例变量必须依存于某一实例，需要先创建对象然后通过对象才能访问到它。
静态变量可以实现让多个对象共享内存。
注：在Java开发中，上下文类和工具类中通常会有大量的静态成员

55.比较一下Java和JavaSciprt

JavaScript与Java是两个公司开发的不同的两个产品。
Java是原SunMicrosystems公司推出的面向对象的程序设计语言，特别适合于互联网应用程序开发；JavaScript是Netscape公司的产品，为了扩展Netscape浏览器的功能而开发的一种可以嵌入Web页面中运行的基于对象和事件驱动的解释性语言。
JavaScript的前身是LiveScript；而Java的前身是Oak语言。
下面对两种语言间的异同作如下比较：
-基于对象和面向对象：Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象；JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object-Based）和事件驱动（Event-Driven）的编程语言，因而它本身提供了非常丰富的内部对象供设计人员使用。
-解释和编译：Java的源代码在执行之前，必须经过编译。JavaScript是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行。（目前的浏览器几乎都使用了JIT（即时编译）技术来提升JavaScript的运行效率）
-强类型变量和类型弱变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中变量是弱类型的，甚至在使用变量前可以不作声明，JavaScript的解释器在运行时检查推断其数据类型。
-代码格式不一样。

56.JDKJREJVM的区别：

Jdk【JavaDevelopmentToolKit】就是java开发工具箱，JDK是整个JAVA的核心里边包含了jre，它除了包含jre之外还包含了一些javac的工具类，把java源文件编译成class文件，java文件是用来运行这个程序的，除此之外，里边还包含了java源生的API，java.lang.integer在rt的jar包里边【可以在项目中看到】，通过rt这个jar包来调用我们的这些io流写入写出等
JDK有以下三种版本：
J2SE，standardedition，标准版，是我们通常用的一个版本
J2EE，enterpsiseedtion，企业版，使用这种JDK开发J2EE应用程序
J2ME，microedtion，主要用于移动设备、嵌入式设备上的java应用程序
Jre【JavaRuntimeEnviromental】是java运行时环境，那么所谓的java运行时环境，就是为了保证java程序能够运行时，所必备的一基础环境，也就是它只是保证java程序运行的，不能用来开发，而jdk才是用来开发的，所有的Java程序都要在JRE下才能运行。
包括JVM和JAVA核心类库和支持文件。与JDK相比，它不包含开发工具—编译器、调试器和其它工具。
Jre里边包含jvm
Jvm：【JavaVirtualMechinal】因为jre是java运行时环境，java运行靠什么运行，而底层就是依赖于jvm，即java虚拟机，java虚拟机用来加载类文件，java中之所以有跨平台的作用，就是因为我们的jvm
关系：
J2se是基于jdk和jre，
JDK是整个JAVA的核心里边包含了jre，
Jre里边包含jvm

57.XML和Json的特点

Xml特点：
1、有且仅有一个根节点；
2、是独立与软件和硬件的信息传输工具（传输量较大）
3、所有的标签都需要自定义
4、仅仅是纯文本文件
Json（JavaScriptObjectNotation）特点：
json分为两种格式:json对象(就是在{}中存储键值对，键和值之间用冒号分隔，键值对之间用逗号分隔)，json数组(就是[]中存储多个json对象，json对象之间用逗号分隔)（两者间可以进行相互嵌套）
数据传输的载体之一
区别：
xml的传输数据量比json的要大，流行的是基于json的数据传输。
共同点：
Xml和json都是传输数据的载体，并且具有跨平台跨语言的特性。

58.JDK常用的包

♦java.lang：这个是系统的基础类，比如String、Math、Integer、System和Thread，提供常用功能。在java.lang包中还有一个子包：java.lang.reflect用于实现java类...
♦java.io:这里面是所有输入输出有关的类，比如文件操作等
♦java.net:这里面是与网络有关的类，比如URL,URLConnection等。
♦java.util:这个是系统辅助类，特别是集合类Collection,List,Map等。
♦java.sql:这个是数据库操作的类，Connection,Statememt，ResultSet等

59.什么是值传递和引用传递?

（1）值传递：形参类型是基本数据类型，方法调用时，实际参数把它的值传递给对应的形式参数，形式参数只是用实际参数的值初始化自己的存储单元内容，是两个不同的存储单元，所以方法执行中形参值得改变不影响实际参数的值
（2）引用传递：形参类型是引用数据类型参数，也称为传地址，方法调用时，实际参数是对象（或数组），这时实际参数与形式参数指向同一个地址，在方法执行中，对形式参数的操作实际上就是对实际参数的操作，这个结果在方法结束后被保留下来，所以方法执行中形式参数的改变将会影响实际参数。

60.解释一下数据库连接池

概念：一种关键的有限的昂贵的资源
影响因素：最大连接数，最小连接数
功能：分配、管理和释放数据库连接
数据库连接池负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而不是再重新建立一个，释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏。这项技术能明显提高对数据库操作的性能。
原理：
连接池基本的思想是在系统初始化的时候，将数据库连接作为对象储在内存中，当用户需要访问数据库时，并非建立一个新的连接，而是从连接池中取出一个已建立的空闲连接对象。使用完毕后，用户也并非将连接关闭，而是将连接放回连接池中，以供下一个请求访问使用。而连接的建立、断开都由连接池自身来管理。同时，还可以通过设置连接池的参数来控制连接池中的初始连接数、连接的上下限数以及每个连接的最大使用次数、最大空闲时间等等。也可以通过其自身的管理机制来监视数据库连接的数量、使用情况。

61.空字符串(“”)和null的区别

空字符串是""，会创建一个对象，内容是""，有内存空间。
而null，不会创建对象，没有内存空间
空字符串是String类的一个对象，而null是指一个引用变量没有引用对象，在值为null的引用变量上调用方法或变量，将会导致NullPointerException。通过以下代码来验证变量x是否为null，x==null，通过”".equals(x)来验证x是否为空字符串。

62.列举java中string类常用方法

- charAt(int index)返回指定索引处的char值。
- concat([String](%22mk:@MSITStore:F:%5C%5C) str)将指定字符串连接到此字符串的结尾。
- equals([Object](%22mk:@MSITStore:F:%5C%5C) anObject)比较字符串的内容是否相等。
- length()返回此字符串的长度。
- trim()返回字符串的副本，忽略前导空白和尾部空白。去掉字符串中的前后空格，中
- substring(int beginIndex)截取子串从beginindex开始。
- substring(int beginIndex,int endIndex)截取子串从哪到哪。
- split([String](%22mk:@MSITStore:F:%5C%5C) regex)分割字符串切割字符串
- valueOf(float f)返回float参数的字符串表示形式。Float可以是任何java提供类型。
- toLowerCase转成小写的与toUpperCase转成大写
- compareTo([String](%22mk:@MSITStore:F:%5C%5C) anotherString)按字典顺序比较两个字符串。
- contains([CharSequence](%22mk:@MSITStore:F:%5C%5C) s)一个字符串中包不包含子串
- indexOf(int ch)返回索引值第一次出现的位置
- replace(char oldChar,char newChar)用newchar替换此字符串中出现的所有

- ◦

63.得到Class的三种方式是什么—反射

反射：是java中的一个机制，也叫反射机制。
反射：是一种Java编程语言的功能。它允许一个运行中的Java程序检查其本身,以及操作程序的内在属性。
第一种：通过每个对象都具备的方法getClass来获取。
第二种：每一个数据类型(基本数据类型和引用数据类型)都有一个静态的属性class。.class用该属性就可以获取到字节码文件对象虽然不用对象调用，但还是要用具体的类调用静态属性。
第三种：使用的Class类中的方法，静态的forName方法。
例：　　Students=newStudent();
　　Classstud=s.getClass();
1种
　　Class<Student>stu=Student.class;　　2种
　　ClassforName=Class.forName("java.util.Data");　　3种
前两种方式不利于程序的扩展，因为都需要在程序使用具体的类来完成。

64.什么是迭代器(Iterator)?

Iterator提供了统一遍历操作集合元素的统一接口,Collection接口实现Iterable接口,每个集合都通过实现Iterable接口中iterator()方法返回Iterator接口的实例,然后对集合的元素进行迭代操作.有一点需要注意的是：在迭代元素的时候不能通过集合的方法删除元素,否则会抛出ConcurrentModificationException异常.但是可以通过Iterator接口中的remove()方法进行删除.

```
-       Iterable接口
    Iterator iterator();
-       Iterator接口
    boolean hasNext();
    Enext();
    void remove();
```

65.char型变量中能不能存贮一个中文汉字?为什么?

char型变量是用来存储Unicode编码的字符的，unicode编码字符集中包含了汉字，所以，char型变量中当然可以存储汉字啦。不过，如果某个特殊的汉字没有被包含在unicode编码字符集中，那么，这个char型变量中就不能存储这个特殊汉字。补充说明：unicode编码占用两个字节，所以，char类型的变量也是占用两个字节。

66.Java中compareTo和compare的区别?

compareTo是Compareable接口的一个方法，主要用于规定创建对象的大小关系，该对象要实现compareable接口，当a.compareTo(b)>0时，则a>b,当a.compareTo(b)<0时，a<b.
compare方法是java.util中的Comparator接口的一个方法，compare方法内主要靠定义compareTo规定的对象大小关系来确定对象的大小。

67.Set里的元素是不能重复的，那么用什么方法来区分重复与否呢?是用==还是equals()?它们有何区别?

Set里的元素是不能重复的，那么用iterator()方法来区分重复与否。equals()是判读两个Set是否相等。
equals()和==方法决定引用值是否指向同一对像equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

68.两个对像值相同(x.equals(y)==true)，但却可有不同的hashcode，这句话对不对?

不对，有相同的hashcode。

69.Java的Socket通信（多线程）

思路：
①首先创建服务器端Socket，指定并侦听某一个端口，然后循环监听开始等待客户端的连接….
②创建客户端socket，指定服务器地址和端口，然后获取输出流，向服务器端发送请求，并关闭socket输出流。
③服务端接收到客户端的请求后，创建新线程并启动。
④创建线程处理类，执行线程操作，获取输入流，服务端读取客户端用户详情，关闭资源。
⑤执行线程操作，获取输出流，响应客户端请求，客户端接受到服务端的响应，关闭资源。
简单点讲，就相当于我跟你说话（客户端→服务端），你收到我说的话（服务端→线程处理类），大脑进行思考后（线程处理类），做出回答我的话（线程处理类→客户端）。

70.接口中定义：

- 静态常量
- 抽象方法

-

71.Break和Continue

Break：退出当前本层循环
Continue：跳过当前这次循环，继续下一次
退出多层循环

72.什么是内部类？StaticNestedClass和InnerClass的不同。

内部类就是定义在另外一个类里面的类!
StaticNestedClass（静态嵌套类）是InnerClass（内部类）的一种，被声明为静态（static）的内部类，因为是静态的，所以嵌套类要创建对象时，不需要外部类的对象，可以以Outer.newInner（）创建对象静态嵌套类不能访问非静态外部类成员，而通常的内部类需要在外部类实例化后才能实例化。

73.下面这条语句一共创建的多少个对象：StringS="a"+"b"+"c"+"d";

代码被编译器编译优化后，相当于直接定义了一个"abcd"的字符串，所以该语句只创建了一个对象

74.什么情况下finally中的代码不会执行?

如果在try块或者catch块中调用了退出虚拟机的方法（即System.exit();）
那么finally中的代码不会执行，不然无论在try块、catch块中执行任何代码，
出现任何情况，异常处理的finally中的代码都要被执行的。

75.一个“.java”源文件中是否可以包括多个类（不是内部类）？有什么限制?

一个“.java”源文件中可以包括多个类（不是内部类）！
单一个文件中只能有一个public类，并且该public类必须与文件名相同

76.Java有没有goto?

goto是Java中的保留字（以后有可能会被启用变成关键字）
Java中的保留字：
byValuecastfururegenericinneroperatorouter
restvarconstgoto

77.问题：如何将String类型转化成Number类型?

Integer类的valueOf方法可以将String转成Number。下面是代码示例：

```
StringnumString=“1000”;
int id=Integer.valueOf(numString).intValue();
```

78.什么是隐式的类型转化?

隐式的类型转化就是简单的一个类型赋值给另一个类型，没有显式的告诉编译器发生了转化。并不是所有的类型都支持隐式的类型转化。

```
int i=1000;
long j=i;//Implicitcasting
```

79.显式的类型转化是什么?

显式的类型转化是明确告诉了编译器来进行对象的转化。
代码示例：

```
long i=700.20;
int j=(int)i;//Explicitcasting
```

80.类型向下转换是什么?

向下转换是指由一个通用类型转换成一个具体的类型，在继承结构上向下进行。

89.如果原地交换两个变量的值?

先把两个值相加赋值给第一个变量，然后用得到的结果减去第二个变量，赋值给第二个变量。再用第一个变量减去第二个变量，同时赋值给第一个变量。代码如下：

```
int a=5,b=10;a=a+b;b=a-b;a=a-b;
```

使用异或操作也可以交换。第一个方法还可能会引起溢出。异或的方法如下：

```
inta=5,b=10;a=a+b;b=a-b;a=a-b;
inta=5;intb=10;
a=a^b;
b=a^b;
a=a^b;
```

90.简述synchronized和java.util.concurrent.locks.Lock的异同?

Lock是Java5以后引入的新的API，和关键字synchronized相比
主要相同点：Lock能完成synchronized所实现的所有功能；
主要不同点：
Syschronized是在jvm层面上的，是一个关键字，而Lock是一个类；
Lock有比synchronized更精确的线程语义和更好的性能，而且不强制性的要求一定要获得锁；
Syschronized同步数据少量的话，性能比Lock好，而数据大量同步，Lock性能要好
synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且最好在finally块中释放（这是释放外部资源的最好的地方）。

91.Thread类的sleep()方法和对象的wait()方法都可以让线程暂停执行，它们有什么区别?

①sleep()方法（休眠）是线程类（Thread）的静态方法，调用此方法会让当前线程暂停执行指定的时间，将执行机会（CPU）让给其他线程，但是对象的锁依然保持，因此休眠时间结束后会自动恢复（线程回到就绪状态。wait()是Object类的方法，调用对象的wait()方法导致当前线程放弃对象的锁（线程暂停执行），进入对象的等待池（waitpool），只有调用对象的notify()方法（或notifyAll()方法）时才能唤醒等
待池中的线程进入等锁池（lockpool），如果线程重新获得对象的锁就可以进入就绪状态。

92.线程的sleep()方法和yield()方法有什么区别?

①sleep()方法给其他线程运行机会时不考虑线程的优先级，因此会给低优先级的线程以运行的机会；yield()方法只会给相同优先级或更高优先级的线程以运行的机会；
②线程执行sleep()方法后转入阻塞（blocked）状态，而执行yield()方法后转入就绪（ready）状态；
③sleep()方法声明抛出InterruptedException，而yield()方法没有声明任何异常；
④sleep()方法比yield()方法（跟操作系统CPU调度相关）具有更好的可移植性。

93.Math.round(11.5)等于多少?Math.round(-11.5)等于多少?

答：Math.round(11.5)==12Math.round(-11.5)==-11round方法返回与参数最接近的长整数，参数加1/2后求其floor。

94.编程题:用最有效率的方法算出2乘以8等於几?

答：2<<3。

95.当一个对象被当作参数传递到方法，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?

答：是值传递。Java编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。

96.定义类A和类B如下

```
class A{

int a=1;

double d=2.0;

void show(){

System.out.println("ClassA:a="+a+"\td="+d);
```

```
}

}

class B extends A{

float a=3.0f;

String d="Javaprogram.";

void show(){

super.show();

System.out.println("ClassB:a="+a+"\td="+d);

}

}
```

(1)若在应用程序的main方法中有以下语句：

```
A a=new A();

a.show();
```

则输出的结果如何?

(2)若在应用程序的main方法中定义类B的对象b：

```
A b=new B();

b.show();
```

则输出的结果如何?

答：输出结果为：

1）ClassA:a=1d=2.0；

2）ClassA:a=1d=2.0

ClassB:a=3.0d=Javaprogram。

97.StaticNestedClass和InnerClass的不同?

答：StaticNestedClass是被声明为静态（static）的内部类，它可以不依赖于外部类实例被实例化。而通常的内部类需要在外部类实例化后才能实例化。

98.abstract的method是否可同时是static,是否可同时是native，是否可同时是synchronized?

答：都不能。

99.静态变量和实例变量的区别?

答：静态变量也称为类变量，归全类共有，它不依赖于某个对象，可通过类名直接访问；而实例变量必须依存于某一实例，只能通过对象才能访问到它。

100.垃圾回收的优点和原理。并考虑2种回收机制?

答：Java语言中一个显著的特点就是引入了垃圾回收机制，使c++程序员最头疼的内存管理的问题迎刃而解，它使得Java程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java中的对象不再有“作用域”的概念，只有对象的引用才有“作用域”。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

101.垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收?

答：对于GC来说，当程序员创建对象时，GC就开始监控这个对象的地址、大小以及使用情况。通常，GC采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是"可达的"，哪些对象是"不可达的"。当GC确定一些对象为"不可达"时，GC就有责任回收这些内存空间。可以。程序员可以手动执行System.gc()，通知GC运行，但是Java语言规范并不保证GC一定会执行。

102.说出一些常用的类，包，接口，请各举5个?

答：常用的类：BufferedReaderBufferedWriterFileReaderFileWirterStringInteger;
常用的包：java.langjava.awtjava.iojava.utiljava.sql;
常用的接口：RemoteListMapDocumentNodeList

103.java中实现多态的机制是什么?

答：方法的覆盖Overriding和重载Overloading是java多态性的不同表现；覆盖Overriding是父类与子类之间多态性的一种表现，重载Overloading是一个类中多态性的一种表现。

104.下面哪些类可以被继承?

1）java.lang.Thread(T)
2）java.lang.Number(T)
3）java.lang.Double(F)
4）java.lang.Math(F)
5）java.lang.Void(F)
6）java.lang.Class(F)
7）java.lang.ClassLoader(T)
答：1、2、7可以被继承。

105.指出下面程序的运行结果

```
class A{

static{

System.out.print("1");

}

public A(){
```

```
System.out.print("2");

}

}

class B extends A{

static{

System.out.print("a");

}

public B(){

System.out.print("b");

}

}

public class Hello{

public static void main(String[]ars){

A ab=new B();//执行到此处,结果:1a2b

ab=new B();//执行到此处,结果:1a2b2b

}

}
```

答：输出结果为1a2b2b；类的static代码段,可以看作是类首次加载(虚拟机加载)执行的代码,而对于类加载,首先要执行其基类的构造,再执行其本身的构造。

106.继承时候类的执行顺序问题,一般都是选择题,问你将会打印出什么?

父类:

```
package test;

public class FatherClass{

public FatherClass(){

 System.out.println("FatherClassCreate");

}

}
```

子类:

```
packa getest;
```

```
import test.FatherClass;

public class ChildClass extends FatherClass{

public ChildClass(){

System.out.println("ChildClassCreate");

}

public static void main(String[]args){

FatherClass fc=new FatherClass();

ChildClass cc=new ChildClass();

}

}
```

答：输出结果为：

```
FatherClassCreate

FatherClassCreate

ChildClassCreate
```

107.关于内部类

```
public class OuterClass{

private double d1=1.0;

//insertcodehere

}

Youneedtoinsertainnerclassdeclarationatline3, Whichtwo

innerclassdeclarationsarevalid?(Choosetwo.)

A.class InnerOne{

public static double method a(){returnd1;}

}

B.public class InnerOne{

static double method a(){returnd1;}

}
```

```
C.private class InnerOne{

double method a(){returnd1;}

}

D.static classInnerOne{

protected double method a(){returnd1;}

}

E.abstract class InnerOne{

public abstract double method a();

}
```

答：答案为C、E；说明如下：

1）静态内部类可以有静态成员，而非静态内部类则不能有静态成员；故A、B错；

2）静态内部类的非静态成员可以访问外部类的静态变量，而不可访问外部类的非静态变量；故D错；

3）非静态内部类的非静态成员可以访问外部类的非静态变量；故C正确。

108.数据类型之间的转换

1)如何将数值型字符转换为数字?
2)如何将数字转换为字符?
3)如何取小数点前两位并四舍五入?
答：1)调用数值类型相应包装类中的方法parse***(String)或valueOf(String)即可返回相应基本类型或包装类型数值；
2)将数字与空字符串相加即可获得其所对应的字符串;另外对于基本类型数字还可调用String类中的valueOf(…)方法返回相应字符串,而对于包装类型数字则可调用其toString()方法获得相应字符串；
3)可用该数字构造一java.math.BigDecimal对象,再利用其round()方法进行四舍五入到保留小数点后两位,再将其转换为字符串截取最后两位。

109.写一个函数，要求输入一个字符串和一个字符长度，对该字符串进行分隔?

答：函数代码如下：

```
public String[] split(Stringstr,intchars){

int n=(str.length()+chars-1)/chars;

String ret[]=newString[n];

for(inti=0;i<n;i++){

if(i<n-1){

ret[i]=str.substring(i*chars,(i+1)*chars);

}else{

ret[i]=str.substring(i*chars);
```

```
}

}

return ret;

}
```

110.写一个函数，2个参数，1个字符串，1个字节数

返回截取的字符串，要求字符串中的中文不能出现乱码：如（“我ABC”，4）应该截为“我AB”，输入（“我ABC汉DEF”，6）应该输出为“我ABC”而不是“我ABC+汉的半个”？

答：代码如下：

```
public String subString(String str,int subBytes){

int bytes=0;//用来存储字符串的总字节数

for(inti=0;i<str.length();i++){

if(bytes==subBytes){

return str.substring(0,i);

}

char c=str.charAt(i);

if(c<256){

bytes+=1;//英文字符的字节数看作1

}else{

bytes+=2;//中文字符的字节数看作2

if(bytes-subBytes==1){

return str.substring(0,i);

}

}

}

return str;

}
```

111.日期和时间?

1)如何取得年月日、小时分秒?
2)如何取得从1970年到现在的毫秒数?
3)如何取得某个日期是当月的最后一天?

4)如何格式化日期?
答：1)创建java.util.Calendar实例(Calendar.getInstance()),调用其get()方法传入不同的参数即可获得参数所对应的值,如：calendar.get(Calendar.YEAR);//获得年
   2)以下方法均可获得该毫秒数:Calendar.getInstance().getTimeInMillis();
System.currentTimeMillis();
   3)示例代码如下：

```
Calendar time=Calendar.getInstance();

time.set(Calendar.DAY_OF_MONTH,

time.getActualMaximum(Calendar.DAY_OF_MONTH));
```

4)利用java.text.DataFormat类中的format()方法可将日期格式化。

112.打印昨天的当前时刻?

```
答：public class YesterdayCurrent{

public static void main(String[]args){

Calendar cal=Calendar.getInstance();

cal.add(Calendar.DATE,-1);

System.out.println(cal.getTime());
}

}
```

113.java和javasciprt的区别?

答：JavaScript与Java是两个公司开发的不同的两个产品。Java是SUN公司推出的新一代面向对象的程序设计语言，特别适合于Internet应用程序开发；而JavaScript是Netscape公司的产品，其目的是为了扩展NetscapeNavigator功能,而开发的一种可以嵌入Web页面中的基于对象和事件驱动的解释性语言,它的前身是LiveScript；而Java的前身是Oak语言。下面对两种语言间的异同作如下比较：

1）基于对象和面向对象：Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象；JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（ObjectBased）和事件驱动（EventDriver）的编程语言。因而它本身提供了非常丰富的内部对象供设计人员使用；

2）解释和编译：Java的源代码在执行之前，必须经过编译；JavaScript是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行；

3）强类型变量和类型弱变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中变量声明，采用其弱类型。即变量在使用前不需作声明，而是解释器在运行时检查其数据类型；

4）代码格式不一样。

114.什么时候用assert?

答：assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。一般来说，assertion用于保证程序最基本、关键的正确性。assertion检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion检查通常是关闭的。在实现中，断言是一个包含布尔表达式的语句，在执行这个语句时假定该表达式为true；如果表达式计算为false，那么系统会报告一个Assertionerror。
断言用于调试目的：
assert(a>0);//throwsanAssertionerrorifa<=0
断言可以有两种形式：
assertExpression1;
assertExpression1:Expression2;
Expression1应该总是产生一个布尔值。
Expression2可以是得出一个值的任意表达式；这个值用于生成显示更多调试信息的String消息。
断言在默认情况下是禁用的，要在编译时启用断言，需使用source1.4标记：
javac-source1.4Test.java
要在运行时启用断言，可使用-enableassertions或者-ea标记。
要在运行时选择禁用断言，可使用-da或者-disableassertions标记。
要在系统类中启用断言，可使用-esa或者-dsa标记。还可以在包的基础上启用或者禁用断言。可以在预计正常情况下不会到达的任何位置上放置断言。断言可以用于验证传递给私有方法的参数。不过，断言不应该用于验证传递给公有方法的参数，因为不管是否启用了断言，公有方法都必须检查其参数。不过，既可以在公有方法中，也可以在非公有方法中利用断言测试后置条件。另外，断言不应该以任何方式改变程序的状态。

115.写出一个你最常见到的runtimeexception?

答：ArithmeticException,ArrayStoreException,BufferOverflowException,

BufferUnderflowException,CannotRedoException,CannotUndoException,

ClassCastException,CMMException,ConcurrentModificationException,

DOMException,EmptyStackException,IllegalArgumentException,

IllegalMonitorStateException,IllegalPathStateException,

IllegalStateException,ImagingOpException,IndexOutOfBoundsException,

MissingResourceException,NegativeArraySizeException,

NoSuchElementException,NullPointerException,ProfileDataException,

ProviderException,RasterFormatException,SecurityException,

SystemException,UndeclaredThrowableException,

UnmodifiableSetException,UnsupportedOperationException

116.类ExampleA继承Exception，类ExampleB继承ExampleA?

有如下代码片断：

try{

throw new ExampleB(“b”);

}catch（ExampleA e）{

System.out.printfln（“ExampleA”）；

```
}catch (Exception e) {

System.out.printfln (“Exception”) ;

}
```

输出的内容应该是：

```
A：ExampleAB：ExceptionC：bD：无
```

答：输出为A。

117.介绍JAVA中的CollectionFrameWork(及如何写自己的数据结构)?

答：CollectionFrameWork如下：

```
Collection

├List

|├LinkedList

|├ArrayList

|└Vector

|└Stack

└Set

Map

├Hashtable

├HashMap

└WeakHashMap
```

Collection是最基本的集合接口，一个Collection代表一组Object，即Collection的元素（Elements）；Map提供key到value的映射。

118.List,Set,Map是否继承自Collection接口?

答：List,Set是；Map不是。

119.你所知道的集合类都有哪些？主要方法?

答：最常用的集合类是List和Map。List的具体实现包括ArrayList和Vector，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象的元素列表。List适用于按数值索引访问元素的情形。Map提供了一个更通用的元素存储方法。Map集合类用于存储元素对（称作“键”和“值”），其中每个键映射到一个值。

120说出ArrayList,Vector,LinkedList的存储性能和特性?

答：ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector由于使用了synchronized方法（线程安全），通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

130.Set里的元素是不能重复的，那么用什么方法来区分重复与否呢?是用==还是equals()?它们有何区别?

答：Set里的元素是不能重复的，用equals()方法来区分重复与否。覆盖equals()方法用来判断对象的内容是否相同，而”==”判断地址是否相等,用来决定引用值是否指向同一对象。

131.用程序给出随便大小的10个数，序号为1-10，按从小到大顺序输出，并输出相应的序号?

答：代码如下：

```
package test;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.List;

import java.util.Random;

public class RandomSort{

public static void printRandomBySort(){

Random random=new Random();//创建随机数生成器

List list=new ArrayList();

//生成10个随机数，并放在集合list中

for(inti=0;i<10;i++){

list.add(random.nextInt(1000));

}

Collections.sort(list);//对集合中的元素进行排序

Iterator it=list.iterator();

int count=0;

while(it.hasNext()){//顺序输出排序后集合中的元素

System.out.println(++count+":"+it.next());

}
```

```
}

public static void main(String[]args){

printRandomBySort();

}

}
```

132.用JAVA实现一种排序，JAVA类实现序列化的方法？在COLLECTION框架中，实现比较要实现什么样的接口?

答：用插入法进行排序代码如下：

```
package test;

import java.util.*;

class InsertSort{

ArrayList al;

public InsertSort(int num,int mod){

al=new ArrayList(num);

Random rand=new Random();

System.out.println("TheArrayListSortBefore:");

for(inti=0;i<num;i++){

al.add(new Integer(Math.abs(rand.nextInt())%mod+

1));

System.out.println("al["+i+"]="+al.get(i));

}

}

public void SortIt(){

tempInt;

int MaxSize=1;

for(inti=1;i<al.size();i++){

tempInt=(Integer)al.remove(i);

if(tempInt.intValue()>=
```

```
((Integer)al.get(MaxSize-1)).intValue()){
al.add(MaxSize,tempInt);
MaxSize++;
System.out.println(al.toString());
}else{
for(intj=0;j<MaxSize;j++){
if(((Integer)al.get(j)).intValue()
\>=tempInt.intValue()){
al.add(j,tempInt);
MaxSize++;
System.out.println(al.toString());
break;
}
}
}
}
System.out.println("TheArrayListSortAfter:");
for(inti=0;i<al.size();i++){
System.out.println("al["+i+"]="+al.get(i));
}
}
public static void main(String[]args){
InsertSort is=new InsertSort(10,100);
is.SortIt();
}
}
```

JAVA类实现序例化的方法是实现java.io.Serializable接口；Collection框架中实现比较要实现Comparable接口和Comparator接口。

133.为什么HashMap链表长度超过8会转成树结构?

HashMap在JDK1.8及以后的版本中引入了红黑树结构，若桶中链表元素个数大于等于8时，链表转换成树结构；若桶中链表元素个数小于等于6时，树结构还原成链表。因为红黑树的平均查找长度是log(n)，长度为8的时候，平均查找长度为3，如果继续使用链表，平均查找长度为8/2=4，这才有转换为树的必要。链表长度如果是小于等于6，6/2=3，虽然速度也很快的，但是转化为树结构和生成树的时间并不会太短。
还有选择6和8，中间有个差值7可以有效防止链表和树频繁转换。假设一下，如果设计成链表个数超过8则链表转换成树结构，链表个数小于8则树结构转换成链表，如果一个HashMap不停的插入、删除元素，链表个数在8左右徘徊，就会频繁的发生树转链表、链表转树，效率会很低。

134.请说出你所知道的线程同步的方法?

答：wait():使一个线程处于等待状态，并且释放所持有的对象的lock；sleep():使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉InterruptedException异常；notify():唤醒一个处于等待状态的线程，注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且不是按优先级；

notityAll():唤醒所有处入等待状态的线程，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争。

135.创建线程的方式?

有三种创建线程的方法：

一是实现Runnable接口，然后将它传递给Thread的构造函数，创建一个Thread对象：

```
new Thread(new Runnable(){

public void run(){

    //重写run方法

    }

    }).start();
```

二是直接继承Thread类：

```
new Thread(){

public void run(){

    //重写run方法

    }

    }.start();
```

三使用Callable和Future创建线程：

```
public class CallableThread implements Callable<String>

{

public static void main(String[]args)

{
```

```
CallableThread ct=new CallableThread();

FutureTask<String>ft=newFutureTask<String>(ct);

System.out.println(ft.get());

}

@Override

public String call()throws Exception

{

return"线程的第三种创建方式";

}

}
```

136.线程同步有几种实现方法,都是什么?

答：java允许多线程并发控制，当多个线程同时操作一个可共享的资源变量时（如数据的增删改查），

将会导致数据不准确，相互之间产生冲突，因此加入同步锁以避免在该线程没有完成操作之前，被其他线程的调用，

从而保证了该变量的唯一性和准确性。

同步的实现方法有五种：

1.同步方法：

有synchronized关键字修饰的方法。

由于java的每个对象都有一个内置锁，当用此关键字修饰方法时，

内置锁会保护整个方法。在调用该方法前，需要获得内置锁，否则就处于阻塞状态。synchronized关键字也可以修饰静态方法，此时如果调用该静态方法，将会锁住整个类

2.同步代码块：

有synchronized关键字修饰的语句块。

被该关键字修饰的语句块会自动被加上内置锁，从而实现同步,同步是一种高开销的操作，因此应该尽量减少同步的内容。

通常没有必要同步整个方法，使用synchronized代码块同步关键代码即可

3.使用特殊域变量(volatile)实现[线程同步](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%90%8C%E6%AD%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PWmvPHDLrjb3uWFbn1uW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH64njDLnHD4):

a.volatile关键字为域变量的访问提供了一种免锁机制，

b.使用volatile修饰域相当于告诉虚拟机该域可能会被其他线程更新

c.因此每次使用该域就要重新计算，而不是使用寄存器中的值

d.volatile不会提供任何原子操作，它也不能用来修饰final类型的变量

4.使用重入锁实现[线程同步](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%90%8C%E6%AD%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PWmvPHDLrjb3uWFbn1uW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH64njDLnHD4)

在JavaSE5.0中新增了一个java.util.concurrent包来支持同步。

ReentrantLock类是可重入、互斥、实现了Lock接口的锁，

它与使用synchronized方法和快具有相同的基本行为和语义，并且扩展了其能力

关于Lock对象和synchronized关键字的选择：

a.最好两个都不用，使用一种java.util.concurrent包提供的机制，

能够帮助用户处理所有与锁相关的代码。

b.如果synchronized关键字能满足用户的需求，就用synchronized，因为它能简化代码

c.如果需要更高级的功能，就用ReentrantLock类，此时要注意及时释放锁，否则会出现死锁，通常在finally代码释放锁

5.使用局部变量实现[线程同步](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%90%8C%E6%AD%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PWmvPHDLrjb3uWFbn1uW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH64njDLnHD4)

使用ThreadLocal管理变量，则每一个使用该变量的线程都获得该变量的副本，

副本之间相互独立，这样每一个线程都可以随意修改自己的变量副本，而不会对其他线程产生影响

137.同步和异步有何异同，在什么情况下分别使用他们?

答：如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

138.启动一个线程是用run()还是start()?

答：启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。

139.线程的基本概念、线程的基本状态以及状态之间的关系?

答：线程指在程序执行过程中，能够执行程序代码的一个执行单位，每个程序至少都有一个线程，也就是程序本身；Java中的线程有四种状态分别是：运行、就绪、挂起、结束。

140.简述synchronized和java.util.concurrent.locks.Lock的异同?

答：主要相同点：Lock能完成synchronized所实现的所有功能；主要不同点：Lock有比synchronized更精确的线程语义和更好的性能。synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且必须在finally从句中释放。

141.用什么关键字修饰同步方法?stop()和suspend()方法为何不推荐使用?

答：用synchronized关键字修饰同步方法；反对使用stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在；suspend()方法容易发生死锁。调用suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被“挂起”的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。故不应该使用suspend()，而应在自己的Thread类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用wait()命其进入等待状态。若标志指出线程应当恢复，则用一个notify()重新启动线程。

142.设计4个线程，其中两个线程每次对j增加1，另两个线程对j每次减少1写出程序?

答：以下程序使用内部类实现线程，对j增减的时候没有考虑顺序问题：

```
public class TestThread{

private int j;

public TestThread(int j ){this.j=j;}

private synchronized void inc(){

j++;

System.out.println(j+"--Inc--"+

Thread.currentThread().getName());

}

private synchronized void dec(){

j--;

System.out.println(j+"--Dec--"+

Thread.currentThread().getName());

}

public void run(){

(new Dec()).start();

new Thread(newInc()).start();

(new Dec()).start();

new Thread(newInc()).start();

}

class Dec extends Thread{
```

```
public void run(){

for(inti=0;i<100;i++){

dec();

}

}

}

class Inc implements Runnable{

public void run(){

for(inti=0;i<100;i++){

inc();

}

}

}

public static void main(String[]args){

(new TestThread(5)).run();

}

}
```

143.什么是java序列化，如何实现java序列化?

答：序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题；序列化的实现：将需要被序列化的类实现Serializable接口，该接口没有需实现的方法，implementsSerializable只是为了标注该对象是可被序列化的，然后使用一个输出流(如FileOutputStream)来构造一个ObjectOutputStream(对象流)对象，接着，使用ObjectOutputStream对象的writeObject(Objectobj)方法就可以将参数为obj的对象写出(即保存其状态)，要恢复的话则用输入流。

144.java中有几种类型的流？JDK为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类?

答：字节流，字符流。字节流继承于InputStream、OutputStream，字符流继承于Reader、Writer。在java.io包中还有许多其他的流，主要是为了提高性能和使用方便。

145.文件和目录（IO）操作?

1)如何列出某个目录下的所有文件?

2)如何列出某个目录下的所有子目录?

3)如何判断一个文件或目录是否存在?

4)如何读写文件?

答：1)示例代码如下：

```
File file=new File("e:\\qianfeng");

File [] files=file.listFiles();

for(int i=0;i<files.length;i++){

if(files[i].isFile())System.out.println(files[i]);

}
```

2)示例代码如下：

```
File file=new File("e:\\qianfeng");

File [] files=file.listFiles();

for(inti=0;i<files.length;i++){

if(files[i].isDirectory())
  System.out.println(files[i]);

}
```

3)创建File对象,调用其exsit()方法即可返回是否存在,如：

```
System.out.println(newFile("d:\\qf.txt").exists());
```

4)示例代码如下：

```
//读文件：

FileInputStream fin=new FileInputStream("e:\\qfjava.txt");

byte[]bs=new byte[100];

while(true){

int len=fin.read(bs);

if(len<=0)
  break;

System.out.print(newString(bs,0,len));

}

fin.close();

//写文件：
```

```
FileWriter fw=new FileWriter("e:\\java.txt");

fw.write("helloworld!"+System.getProperty("line.separator"));

fw.write("XXX公司，JAVA欢迎你");

fw.close();
```

146.写一个方法,输入一个文件名和一个字符串,统计这个字符串在这个文件中出现的次数?

答：代码如下：

```
public int countWords(String file,String find)throws Exception

{

int count=0;

Reader in=new FileReader(file);

int c;

while((c=in.read())!=-1){

while(c==find.charAt(0)){

for(inti=1;i<find.length();i++){

c=in.read();

if(c!=find.charAt(i))break;

if(i==find.length()-1)count++;

}

}

}

return count;

}
```

147.用JAVASOCKET编程，读服务器几个字符，再写入本地显示?

答：Server端程序：

```
package test;

import java.net.*;

import java.io.*;

public class Server{
```

```
private ServerSocket ss;

private Socketsocket;

private BufferedReader in;

private PrintWriter out;

public Server(){

try{

ss=new ServerSocket(10000);

while(true){

socket=ss.accept();

String RemoteIP=

socket.getInetAddress().getHostAddress();

String RemotePort=":"+socket.getLocalPort();

System.out.println("Aclientcomein!IP:"

+RemoteIP+RemotePort);

in=newBufferedReader(new

InputStreamReader(socket.getInputStream()));

String line=in.readLine();

System.out.println("Cleintsendis:"+line);

out=

new PrintWriter(socket.getOutputStream(),true);

out.println("YourMessageReceived!");

out.close();

in.close();

socket.close();

}

}catch(IOExceptione){

out.println("wrong");

}

}
```

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

new Server();

}

}
```

Client端程序:

```
package test;

import java.io.*;

import java.net.*;

public class Client{

Socket socket;

BufferedReader in;

PrintWriter out;

public Client(){

try{

System.out.println("TrytoConnectto

127.0.0.1:10000");

socket=new Socket("127.0.0.1",10000);

System.out.println("TheServerConnected!");

System.out.println("PleaseentersomeCharacter:");

BufferedReader line=new BufferedReader(new InputStreamReader(System.in));

out=new PrintWriter(socket.getOutputStream(),true);

out.println(line.readLine());

in=new BufferedReader(

new InputStreamReader(socket.getInputStream()));

System.out.println(in.readLine());

out.close();

in.close();

socket.close();
```

```
}catch(IOExceptione){

out.println("Wrong");

}

}

public static void main(String[]args){

new Client();

}

}
```

148.什么是设计模式？分类是什么？

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

其实还有两类：并发型模式和线程池模式。

149.设计模式的六大原则？

1、开闭原则（OpenClosePrinciple）

开闭原则就是说对扩展开放，对修改关闭。在程序需要进行拓展的时候，不能去修改原有的代码，实现一个热插拔的效果。所以一句话概括就是：为了使程序的扩展性好，易于维护和升级。想要达到这样的效果，我们需要使用接口和抽象类，后面的具体设计中我们会提到这点。

2、里氏代换原则（LiskovSubstitutionPrinciple）

里氏代换原则(LiskovSubstitutionPrincipleLSP)面向对象设计的基本原则之一。里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。LSP是继承复用的基石，只有当衍生类可以替换掉基类，软件单位的功能不受到影响时，基类才能真正被复用，而衍生类也能够在基类的基础上增加新的行为。里氏代换原则是对“开-闭”原则的补充。实现“开-闭”原则的关键步骤就是抽象化。而基类与子类的继承关系就是抽象化的具体实现，所以里氏代换原则是对实现抽象化的具体步骤的规范。—— FromBaidu百科

3、依赖倒转原则（DependenceInversionPrinciple）

这个是开闭原则的基础，具体内容：真对接口编程，依赖于抽象而不依赖于具体。

4、接口隔离原则（InterfaceSegregationPrinciple）

这个原则的意思是：使用多个隔离的接口，比使用单个接口要好。还是一个降低类之间的耦合度的意思，从这儿我们看出，其实设计模式就是一个软件的设计思想，从大型软件[架构](http://lib.csdn.net/base/architecture%22%20%5Co)出发，为了升级和维护方便。所以上文中多次出现：降低依赖，降低耦合。

5、迪米特法则（最少知道原则）（DemeterPrinciple）

为什么叫最少知道原则，就是说：一个实体应当尽量少的与其他实体之间发生相互作用，使得系统功能模块相对独立。

6、合成复用原则（CompositeReusePrinciple）

原则是尽量使用合成/聚合的方式，而不是使用继承。

150.写一个Singleton出来?

答：Singleton模式主要作用是保证在Java应用程序中，一个类Class只有一个实例存在。举例：定义一个类，它的构造函数为private的，它有一个static的private的该类变量，在类初始化时实例话，通过一个public的getInstance

方法获取对它的引用,继而调用其中的方法。

第一种形式：

```
public class Singleton{

private Singleton(){}

private static Singleton instance=new Singleton();

public static Singleton getInstance(){

return instance;

}

}
```

第二种形式：

```
public class Singleton{

private static Singleton instance=null;

public static synchronized Singleton getInstance(){

if(instance==null)

instance＝new Singleton();

return instance;

}

}
```

其他形式:定义一个类，它的构造函数为private的，所有方法为static的。一般认为第一种形式要更加安全些。

151.说说你所熟悉或听说过的JAVAEE中的几种常用模式?及对设计模式的一些看法?

答：SessionFacadePattern：使用SessionBean访问EntityBean；

MessageFacadePattern：实现异步调用；

EJBCommandPattern：使用CommandJavaBeans取代SessionBean，实现轻量级访问；

DataTransferObjectFactory：通过DTOFactory简化EntityBean数据提供特性；

GenericAttributeAccess：通过AttibuteAccess接口简化EntityBean数据提供特性；

BusinessInterface：通过远程（本地）接口和Bean类实现相同接口规范业务逻辑一致性；

152.Java中常用的设计模式？说明工厂模式？【中等难度】

答：Java中的23种设计模式：Factory（工厂模式），Builder（建造模式），FactoryMethod（工厂方法模式），Prototype（原始模型模式），Singleton（单例模式），Facade（门面模式），Adapter（适配器模式），Bridge（桥梁模式），Composite（合成模式），Decorator（装饰模式），Flyweight（享元模式），Proxy（代理模式），Command（命令模式），Interpreter（解释器模式），Visitor（访问者模式），Iterator（迭代子模式），Mediator（调停者模式），Memento（备忘录模式），Observer（观察者模式），State（状态模式），Strategy（策略模式），TemplateMethod（模板方法模式），ChainOfResponsibleity（责任链模式）。

工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某一个类的实例，通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

153.开发中都用到了那些设计模式?用在什么场合?

答：每个模式都描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心。通过这种方式，你可以无数次地使用那些已有的解决方案，无需在重复相同的工作。主要用到了MVC的设计模式，用来开发JSP/Servlet或者J2EE的相关应用；及简单工厂模式等。154.你对软件开发中迭代的含义的理解?

答：软件开发中，各个开发阶段不是顺序执行的，应该是并行执行,也就是迭代的意思。这样对于开发中的需求变化，及人员变动都能得到更好的适应。

155.你经常读那些书?

Java编程思想Java模式人月神话.....

156.git,svn区别

Git是分布式的，SVN是集中式的，好处是跟其他同事不会有太多的冲突，自己写的代码放在自己电脑上，一段时间后再提交、合并，也可以不用联网在本地提交；

157.XML文档定义有几种形式？它们之间有何本质区别？解析XML文档有哪几种方式?

答：1）两种形式：dtd以及schema；

2）本质区别：schema本身是xml的，可以被XML解析器解析(这也是从DTD上发展schema的根本目的)；

3）解析方式：有DOM,SAX,STAX、DOM4J等：

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由DOM的树结构所造成的，这种结构占用的内存较多，而且DOM必须在解析文件之前把整个文档装入内存,适合对XML的随机访问；SAX:不同于DOM,SAX是事件驱动型的XML解析方式。它顺序读取XML文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理XML文件，适合对XML的顺序访问；

STAX:StreamingAPIforXML(StAX)。

158.你在项目中用到了xml技术的哪些方面?如何实现的?

答:用到了数据存贮，信息配置两方面。在做数据交换平台时，将不能数据源的数据组装成XML文件，然后将XML文件压缩打包加密后通过网络传送给接收者，接收解密与解压缩后再同XML文件中还原相关信息进行处理。在做软件配置时，利用XML可以很方便的进行，软件的各种配置参数都存贮在XML文件中。

159.什么是重量级？什么是轻量级?

答：轻量级是指它的创建和销毁不需要消耗太多的资源，意味着可以在程序中经常创建和销毁session的对象；重量级意味不能随意的创建和销毁它的实例，会占用很多的资源。

160.为什么要用clone?

在实际编程过程中，我们常常要遇到这种情况：有一个对象A，在某一时刻A中已经包含了一些有效值，此时可能会需要一个和A完全相同新对象B，并且此后对B任何改动都不会影响到A中的值，也就是说，A与B是两个独立的对象，但B的初始值是由A对象确定的。在Java语言中，用简单的赋值语句是不能满足这种需求的。要满足这种需求虽然有很多途径，但实现clone（）方法是其中最简单，也是最高效的手段。

161.new一个对象的过程和clone一个对象的过程区别

new操作符的本意是分配内存。程序执行到new操作符时，首先去看new操作符后面的类型，因为知道了类型，才能知道要分配多大的内存空间。分配完内存之后，再调用构造函数，填充对象的各个域，这一步叫做对象的初始化，

构造方法返回后，一个对象创建完毕，可以把他的引用（地址）发布到外部，在外部就可以使用这个引用操纵这个对象。

clone在第一步是和new相似的，都是分配内存，调用clone方法时，分配的内存和原对象（即调用clone方法的对象）相同，然后再使用原对象中对应的各个域，填充新对象的域，填充完成之后，clone方法返回，一个新的相同的对象被创建，同样可以把这个新对象的引用发布到外部。

161.复制对象和复制引用的区别

```
Person p=new Person(23,"zhang");

Person p1=p;

System.out.println(p);

System.out.println(p1);
```

当Person p1=p;执行之后，是创建了一个新的对象吗？首先看打印结果：

1. com.qf.Person@2f9ee1ac

2. com.qf.Person@2f9ee1ac

可以看出，打印的地址值是相同的，既然地址都是相同的，那么肯定是同一个对象。p和p1只是引用而已，他们都指向了一个相同的对象Person(23,“zhang”)。可以把这种现象叫做引用的复制。上面代码执行完成之后，内存中的情景如下图所示：

而下面的代码是真真正正的克隆了一个对象。

1. Person p=new Person(23,"zhang");

2. Person p1=(Person)p.clone();3. System.out.println(p);

4. System.out.println(p1);

从打印结果可以看出，两个对象的地址是不同的，也就是说创建了新的对象，而不是把原对象的地址赋给了一个新的引用变量：

1.com.qf.Person@2f9ee1ac

2.com.qf.Person@67f1fba0

162.深拷贝和浅拷贝

上面的示例代码中，Person中有两个成员变量，分别是name和age，name是String类型，age是int类型。代码非常简单，如下所示：

由于age是基本数据类型，那么对它的拷贝没有什么意义，直接将一个4字节的整数值拷贝过来就行。但是name是String类型的，它只是一个引用，指向一个真正的String对象，那么对它的拷贝有两种方式：直接将原对象中的name的引用值拷贝给新对象的name字段，或者是根据原Person对象中的name指向的字符串对象创建一个新的相同的字符串对象，将这个新字符串对象的引用赋给新拷贝的Person对象的name字段。这两种拷贝方式分别叫做浅拷贝和深拷贝。深拷贝和浅拷贝的原理如下图所示：

下面通过代码进行验证。如果两个Person对象的name的地址值相同，说明两个对象的name都指向同一个String对象，也就是浅拷贝，而如果两个对象的name的地址值不同，那么就说明指向不同的String对象，也就是在拷贝Person对象的时候，同时拷贝了name引用的String对象，也就是深拷贝。验证代码如下：

打印结果为：clone是浅拷贝的。

所以，clone方法执行的是浅拷贝，在编写程序时要注意这个细节。

如何进行深拷贝：

由上内容可以得出如下结论：如果想要深拷贝一个对象，这个对象必须要实现Cloneable接口，实现clone

方法，并且在clone方法内部，把该对象引用的其他对象也要clone一份，这就要求这个被引用的对象必须也要实现

Cloneable接口并且实现clone方法。那么，按照上面的结论，实现以下代码Body类组合了Head类，要想深拷贝

Body类，必须在Body类的clone方法中将Head类也要拷贝一份。代码如下：

打印结果为：

body==body1:false

body.head==body1.head:false

163.两个对象值相同(x.equals(y)==true)，但却可有不同的hashCode，这句话对不对?

不对，如果两个对象x和y满足x.equals(y)==true，它们的哈希码（hashCode）应当相同。

Java对于eqauls方法和hashCode方法是这样规定的：(1)如果两个对象相同（equals方法返回true），那么它们的hashCode值一定要相同；(2)如果两个对象的hashCode相同，它们并不一定相同。当然，你未必要按照要求去做，但是如果你违背了上述原则就会发现在使用容器时，相同的对象可以出现在Set集合中，同时增加新元素的效率会大大下降（对于使用哈希存储的系统，如果哈希码频繁的冲突将会造成存取性能急剧下降）。

关于equals和hashCode方法，很多Java程序员都知道，但很多人也就是仅仅知道而已，在JoshuaBloch的大作《EffectiveJava》（很多软件公司，《EffectiveJava》、《Java编程思想》以及《重构：改善既有代码质量》是Java程序员必看书籍，如果你还没看过，那就赶紧去买一本吧）中是这样介绍equals方法的。

首先equals方法必须满足自反性（x.equals(x)必须返回true）、对称性（x.equals(y)返回true时，y.equals(x)也必须返回true）、传递性（x.equals(y)和y.equals(z)都返回true时，x.equals(z)也必须返回true）和一致性（当

x和y引用的对象信息没有被修改时，多次调用x.equals(y)应该得到同样的返回值），而且对于任何非null值的引用x，x.equals(null)必须返回false。

实现高质量的equals方法的诀窍包括：

1. 使用==操作符检查"参数是否为这个对象的引用"；

2.使用instanceof操作符检查"参数是否为正确的类型"；

3.对于类中的关键属性，检查参数传入对象的属性是否与之相匹配；

4.编写完equals方法后，问自己它是否满足对称性、传递性、一致性；

5.重写equals时总是要重写hashCode；

6.不要将equals方法参数中的Object对象替换为其他的类型，在重写时不要忘掉@Override注解。

164.当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?

是值传递。Java语言的方法调用只支持参数的值传递。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的属性可以在被调用过程中被改变，但对对象引用的改变是不会影响到调用者的。C++和C#中可以通过传引用或传输出参数来改变传入的参数的值。说明：Java中没有传引用实在是非常的不方便，这一点在Java8中仍然没有得到改进，正是如此在Java编写的代码中才会出现大量的Wrapper类（将需要通过方法调用修改的引用置于一个Wrapper类中，再将Wrapper对象传入方法），这样的做法只会让代码变得臃肿，尤其是让从C和C++转型为Java程序员的开发者无法容忍。

165.为什么函数不能根据返回类型来区分重载?

该道题来自华为面试题。

因为调用时不能指定类型信息，编译器不知道你要调用哪个函数。例如：

当调用max(1,2);时无法确定调用的是哪个，单从这一点上来说，仅返回值类型不同的重载是不应该允许的。再比如对下面这两个方法来说，虽然它们有同样的名字和自变量，但其实是很容易区分的：

若编译器可根据上下文（语境）明确判断出含义，比如在intx=f()中，那么这样做完全没有问题。然而，我们也可能调用一个方法，同时忽略返回值；我们通常把这称为“为它的副作用去调用一个方法”，因为我

们关心的不是返回值，而是方法调用的其他效果。所以假如我们像下面这样调用方法：f();Java怎样判断f()的具体调用方式呢？而且别人如何识别并理解代码呢？由于存在这一类的问题，所以不能。

函数的返回值只是作为函数运行之后的一个“状态”，他是保持方法的调用者与被调用者进行通信的关键。并不能作为某个方法的“标识”。

165.char型变量中能不能存储一个中文汉字，为什么?

char类型可以存储一个中文汉字，因为Java中使用的编码是Unicode（不选择任何特定的编码，直接

使用字符在字符集中的编号，这是统一的唯一方法），一个char类型占2个字节（16比特），所以放一个中文是没问题的。

补充：使用Unicode意味着字符在JVM内部和外部有不同的表现形式，在JVM内部都是Unicode，当这个字符被从JVM内部转移到外部时（例如存入文件系统中），需要进行编码转换。所以Java中有字节流和字符流，以及在字符流和字节流之间进行转换的转换流，如InputStreamReader和OutputStreamReader，这两个类是字节流和字符流之间的适配器类，承担了编码转换的任务；对于C程序员来说，要完成这样的编码转换恐怕要依赖于union（联合体/共用体）共享内存的特征来实现了。

166.请说出下面程序的输出

167.如何取得年月日、小时分钟秒?

168.如何取得从1970年1月1日0时0分0秒到现在的毫秒数?

169.如何取得某月的最后一天?

170. 如何格式化日期?

1）Java.text.DataFormat的子类（如SimpleDateFormat类）中的format(Date)方法可将日期格式化。

2）Java8中可以用java.time.format.DateTimeFormatter来格式化时间日期，代码如下所示：

补充：Java的时间日期API一直以来都是被诟病的东西，为了解决这一问题，Java8中引入了新的时间日期API，其中包括LocalDate、LocalTime、LocalDateTime、Clock、Instant等类，这些的类的设计都使用了不变模式，因此是线程安全的设计。

171. int和Integer有什么区别?

Java是一个近乎纯洁的面向对象编程语言，但是为了编程的方便还是引入了基本数据类型，为了能够将这些基本数据类型当成对象操作，Java为每一个基本数据类型都引入了对应的包装类型（wrapperclass），int的包装类就是Integer，从Java5开始引入了自动装箱/拆箱机制，使得二者可以相互转换。

Java为每个原始类型提供了包装类型：

原始类型:boolean，char，byte，short，int，long，float，double

包装类型：Boolean，Character，Byte，Short，Integer，Long，Float，Doubl

172. 下面Integer类型的数值比较输出的结果为?

如果不明就里很容易认为两个输出要么都是true要么都是false。首先需要注意的是f1、f2、f3、f4四个变量都是Integer对象引用，所以下面的==运算比较的不是值而是引用。装箱的本质是什么呢？当我们给一个Integer对象赋一个int值的时候，会调用Integer类的静态方法valueOf，如果看看valueOf的源代码就知道发生了什么。

源码

IntegerCache是Integer的内部类，其代码如下所示：

简单的说，如果整型字面量的值在-128到127之间，那么不会new新的Integer对象，而是直接引用常量池中的Integer对象，所以上面的面试题中f1==f2的结果是true，而f3==f4的结果是false。

173. String类常用方法

174. Java中有几种类型的流

按照流的方向：输入流（inputStream）和输出流（outputStream）。

按照实现功能分：节点流（可以从或向一个特定的地方（节点）读写数据。如FileReader）和处理流（是对一个已存在的流的连接和封装，通过所封装的流的功能调用实现数据读写。如BufferedReader。处理流的构造方法总是要带一个其他的流对象做参数。一个流对象经过其他流的多次包装，称为流的链接。）

按照处理数据的单位：字节流和字符流。字节流继承于InputStream和OutputStream，字符流继承于

InputStreamReader和OutputStreamWriter。

175. 字节流如何转为字符流

字节输入流转字符输入流通过InputStreamReader实现，该类的构造函数可以传入InputStream对象。

字节输出流转字符输出流通过OutputStreamWriter实现，该类的构造函数可以传入OutputStream对象。

176. 如何将一个java对象序列化到文件里

在java中能够被序列化的类必须先实现Serializable接口，该接口没有任何抽象方法只是起到一个标记作用。

177. 字节流和字符流的区别

字节流读取的时候，读到一个字节就返回一个字节；字符流使用了字节流读到一个或多个字节（中文对应的字节数是两个，在UTF-8码表中是3个字节）时。先去查指定的编码表，将查到的字符返回。字节流可以处理所有类型数据，如：图片，MP3，AVI视频文件，而字符流只能处理字符数据。只要是处理纯文本数据，就要优先考虑使用字符流，除此之外都用字节流。字节流主要是操作byte类型数据，以byte数组为准，主要操作类就是OutputStream、InputStream

字符流处理的单元为2个字节的Unicode字符，分别操作字符、字符数组或字符串，而字节流处理单元为1个字节，操作字节和字节数组。所以字符流是由Java虚拟机将字节转化为2个字节的Unicode字符为单位的字符而成的，所以它对多国语言支持性比较好！如果是音频文件、图片、歌曲，就用字节流好点，如果是关系到中文（文本）的，用字符流好点。在程序中一个字符等于两个字节，java提供了Reader、Writer两个专门操作字符流的类。

178. 什么是java序列化，如何实现java序列化?

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题。

序列化的实现：将需要被序列化的类实现Serializable接口，该接口没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化的，然后使用一个输出流(如：FileOutputStream)来构造一个ObjectOutputStream(对象流)对象，接着，使用ObjectOutputStream对象的writeObject(Objectobj)方法就可以将参数为obj的对象写出(即保存其状态)，要恢复的话则用输入流。

179. 集合的安全性问题

请问ArrayList、HashSet、HashMap是线程安全的吗？如果不是我想要线程安全的集合怎么办?

我们都看过上面那些集合的源码（如果没有那就看看吧），每个方法都没有加锁，显然都是线程不安全的。话又说过来如果他们安全了也就没第二问了。

在集合中Vector和HashTable倒是线程安全的。你打开源码会发现其实就是把各自核心方法添加上了synchronized关键字。

Collections工具类提供了相关的API，可以让上面那3个不安全的集合变为安全的。

```
1.      Collections.synchronizedCollection(c)

2.      Collections.synchronizedList(list)

3.      Collections.synchronizedMap(m)

4.      Collections.synchronizedSet(s)
```

上面几个函数都有对应的返回值类型，传入什么类型返回什么类型。打开源码其实实现原理非常简单，就是将集合的核心方法添加上了synchronized关键字。

180. ArrayList内部用什么实现的?

（回答这样的问题，不要只回答个皮毛，可以再介绍一下ArrayList内部是如何实现数组的增加和删除的，因为数组在创建的时候长度是固定的，那么就有个问题我们往ArrayList中不断的添加对象，它是如何管理这些数组呢？）

ArrayList内部是用Object[]实现的。接下来我们分别分析ArrayList的构造、add、remove、clear方法的实现原理。

一、构造函数

1）空参构造

array是一个Object[]类型。当我们new一个空参构造时系统调用了EmptyArray.OBJECT属性，EmptyArray仅仅是一个系统

的类库，该类源码如下：

也就是说当我们new一个空参ArrayList的时候，系统内部使用了一个newObject[0]数组。

2）带参构造1

```
/**

*Constructsanewinstanceof{@codeArrayList}withthespecified

*initialcapacity.

*

*@paramcapacity

*theinitialcapacityofthis{@codeArrayList}.

*/

public ArrayList(intcapacity){if(capacity<0){

throw new IllegalArgumentException("capacity<0:"+capacity);

}

array=(capacity==0?EmptyArray.OBJECT:newObject[capacity]);

}
```

该构造函数传入一个int值，该值作为数组的长度值。如果该值小于0，则抛出一个运行时异常。如果等于0，则使用一个空数组，如果大于0，则创建一个长度为该值的新数组。

3）带参构造2

```
/**

*Constructsanewinstanceof{@codeArrayList}containingtheelementsof

*thespecifiedcollection.

*

*@paramcollection

*thecollectionofelementstoadd.

*/

public ArrayList(Collection<?extendsE>collection){if(collection==null){

throw new NullPointerException("collection==null");

}

Object[]a=collection.toArray();if(a.getClass()!=Object[].class){
```

```
Object[]newArray=newObject[a.length];System.arraycopy(a,0,newArray,0,a.length);a=newArray;

}

array=a;

size=a.length;

}
```

如果调用构造函数的时候传入了一个Collection的子类，那么先判断该集合是否为null，为null则抛出空指针异常。如果不是则将该集合转换为数组a，然后将该数组赋值为成员变量array，将该数组的长度作为成员变量size。这里面它先判断a.getClass是否等于Object[].class，其实一般都是相等的，我也暂时没想明白为什么多加了这个判断，

toArray方法是Collection接口定义的，因此其所有的子类都有这样的方法，list集合的toArray和Set集合的toArray

返回的都是Object[]数组。

这里讲些题外话，其实在看Java源码的时候，作者的很多意图都很费人心思，我能知道他的目标是啥，但是不知道他为何这样写。比如对于ArrayList，array是他的成员变量，但是每次在方法中使用该成员变量的时候作者都会重新在方法中开辟一个局部变量，然后给局部变量赋值为array，然后再使用，有人可能说这是为了防止并发修改array，毕竟array是成员变量，大家都可以使用因此需要将array变为局部变量，然后再使用，这样的说法并不是都成立的，也许有时候就是老外们写代码的一个习惯而已。

二、add方法

add方法有两个重载，这里只研究最简单的那个。

```
/**

*Addsthespecifiedobjectattheendofthis{@codeArrayList}.

*

*@paramobject

*theobjecttoadd.

*@returnalwaystrue

*/

@Overridepublicbooleanadd(Eobject){Object[]a=array;

ints=size;

if(s==a.length){

Object[]newArray=newObject[s+

(s<(MIN_CAPACITY_INCREMENT/2)?MIN_CAPACITY_INCREMENT:s>>1)];

System.arraycopy(a,0,newArray,0,s);array=a=newArray;

}

a[s]=object;size=s+1;modCount++;returntrue;
```

```
}
```

1、首先将成员变量array赋值给局部变量a，将成员变量size赋值给局部变量s。

2、判断集合的长度s是否等于数组的长度（如果集合的长度已经等于数组的长度了，说明数组已经满了，该重新分配新数组了），重新分配数组的时候需要计算新分配内存的空间大小，如果当前的长度小于

MIN_CAPACITY_INCREMENT/2（这个常量值是12，除以2就是6，也就是如果当前集合长度小于6）则分配12个

长度，如果集合长度大于6则分配当前长度s的一半长度。这里面用到了三元运算符和位运算，s>>1，意思就是将

s往右移1位，相当于s=s/2，只不过位运算是效率最高的运算。

3、将新添加的object对象作为数组的a[s]个元素。4、修改集合长度size为s+1

5、modCotun++,该变量是父类中声明的，用于记录集合修改的次数，记录集合修改的次数是为了防止在用迭代器迭代集合时避免并发修改异常，或者说用于判断是否出现并发修改异常的。

6、returntrue，这个返回值意义不大，因为一直返回true，除非报了一个运行时异常。三、remove方法

remove方法有两个重载，我们只研究remove（intindex）方法。

```
/**

*Removestheobjectatthespecifiedlocationfromthislist.

*

*@paramindex

*theindexoftheobjecttoremove.

*@returntheremovedobject.

*@throwsIndexOutOfBoundsException

*when{@codelocation<0||location>=size()}

*/

@OverridepublicEremove(intindex){Object[]a=array;

ints=size;

if(index>=s){throw IndexOutOfBoundsException(index,s);

}

@SuppressWarnings("unchecked")Eresult=(E)a[index];

System.arraycopy(a,index+1,a,index,--s-index);

a[s]=null;//Preventmemoryleak

size=s;modCount++;returnresult;
```

}

1、先将成员变量array和size赋值给局部变量a和s。

2、判断形参index是否大于等于集合的长度，如果成了则抛出运行时异常

3、获取数组中脚标为index的对象result，该对象作为方法的返回值

4、调用System的arraycopy函数，拷贝原理如下图所示。

5、接下来就是很重要的一个工作，因为删除了一个元素，而且集合整体向前移动了一位，因此需要将集合最后一个元素设置为null，否则就可能内存泄露。

6、重新给成员变量array和size赋值

7、记录修改次数

8、返回删除的元素（让用户再看最后一眼）四、clear方法

如果集合长度不等于0，则将所有数组的值都设置为null，然后将成员变量size设置为0即可，最后让修改记录加1。

181. 并发集合和普通集合如何区别?

并发集合常见的有ConcurrentHashMap、ConcurrentLinkedQueue、ConcurrentLinkedDeque等。并发集合位于java.util.concurrent包下，是jdk1.5之后才有的，主要作者是DougLea

（http://baike.baidu.com/view/3141057.htm）完成的。

在java中有普通集合、同步（线程安全）的集合、并发集合。普通集合通常性能最高，但是不保证多线程的安全性和并发的可靠性。线程安全集合仅仅是给集合添加了synchronized同步锁，严重牺牲了性能，而且对并发的效率就更低了，并发集合则通过复杂的策略不仅保证了多线程的安全又提高的并发时的效率。

参考阅读:

ConcurrentHashMap是线程安全的HashMap的实现，默认构造同样有initialCapacity和loadFactor属性，不过还多了一个concurrencyLevel属性，三属性默认值分别为16、0.75及16。其内部使用锁分段技术，维持这锁Segment的数组，在Segment数组中又存放着Entity[]数组，内部hash算法将数据较均匀分布在不同锁中。

put操作：并没有在此方法上加上synchronized，首先对key.hashcode进行hash操作，得到key的hash值。hash操作的算法和map也不同，根据此hash值计算并获取其对应的数组中的Segment对象(继承自ReentrantLock)，接着调用此Segment对象的put方法来完成当前操作。

ConcurrentHashMap基于concurrencyLevel划分出了多个Segment来对key-value进行存储，从而避免每次put操作都得锁住整个数组。在默认的情况下，最佳情况下可允许16个线程并发无阻塞的操作集合对象，尽可能地减少并发时的阻塞现象。

get(key)

首先对key.hashCode进行hash操作，基于其值找到对应的Segment对象，调用其get方法完成当前操作。而Segment的get操作首先通过hash值和对象数组大小减1的值进行按位与操作来获取数组上对应位置的

HashEntry。在这个步骤中，可能会因为对象数组大小的改变，以及数组上对应位置的HashEntry产生不一致性，那么ConcurrentHashMap是如何保证的?

对象数组大小的改变只有在put操作时有可能发生，由于HashEntry对象数组对应的变量是volatile类型的，因此可以保证如HashEntry对象数组大小发生改变，读操作可看到最新的对象数组大小。

在获取到了HashEntry对象后，怎么能保证它及其next属性构成的链表上的对象不会改变呢？这点ConcurrentHashMap采用了一个简单的方式，即HashEntry对象中的hash、key、next属性都是final的，这也就意味着没办法插入一个HashEntry对象到基于next属性构成的链表中间或末尾。这样就可以保证当获取到HashEntry对象后，其基于next属性构建的链表是不会发生变化的。

ConcurrentHashMap默认情况下采用将数据分为16个段进行存储，并且16个段分别持有各自不同的锁

Segment，锁仅用于put和remove等改变集合对象的操作，基于volatile及HashEntry链表的不变性实现了读取的不加锁。这些方式使得ConcurrentHashMap能够保持极好的并发支持，尤其是对于读远比插入和删除频繁的Map而言，而它采用的这些方法也可谓是对于Java内存模型、并发机制深刻掌握的体现。

182. List的三个子类的特点

ArrayList底层结构是数组,底层查询快,增删慢。

LinkedList底层结构是链表型的,增删快,查询慢。

voctor　底层结构是数组线程安全的,增删慢,查询慢。

183. List和Map、Set的区别

结构特点

List和Set是存储单列数据的集合，Map是存储键和值这样的双列数据的集合；List中存储的数据是有顺序，并且允许重复；Map中存储的数据是没有顺序的，其键是不能重复的，它的值是可以有重复的，Set中存储的数据是无序的，且不允许有重复，但元素在集合中的位置由元素的hashcode决定，位置是固定的（Set集合根据hashcode来进行数据的存储，所以位置是固定的，但是位置不是用户可以控制的，所以对于用户来说set中的元素还是无序的）；

实现类

List接口有三个实现类（LinkedList：基于链表实现，链表内存是散乱的，每一个元素存储本身内存地址的同时还存储下一个元素的地址。链表增删快，查找慢；ArrayList：基于数组实现，非线程安全的，效率高，便于索引，但不便于插入删除；Vector：基于数组实现，线程安全的，效率低）。

Map接口有三个实现类（HashMap：基于hash表的Map接口实现，非线程安全，高效，支持null值和null键；HashTable：线程安全，低效，不支持null值和null键；LinkedHashMap：是HashMap的一个子类，保存了记录的插入顺序；SortMap接口：TreeMap，能够把它保存的记录根据键排序，默认是键值的升序排序）。

Set接口有两个实现类（HashSet：底层是由HashMap实现，不允许集合中有重复的值，使用该方式时需要重写equals()和hashCode()方法；LinkedHashSet：继承与HashSet，同时又基于LinkedHashMap来进行实现，底层使用的是LinkedHashMp）。

区别

List集合中对象按照索引位置排序，可以有重复对象，允许按照对象在集合中的索引位置检索对象，例如通过list.get(i)方法来获取集合中的元素；Map中的每一个元素包含一个键和一个值，成对出现，键对象不可以重复，值对象可以重复；Set集合中的对象不按照特定的方式排序，并且没有重复对象，但它的实现类能对集合中的对象按照特定的方式排序，例如TreeSet类，可以按照默认顺序，也可以通过实现Java.util.Comparator<Type>接口来自定义排序方式。

184. 数组和链表的区别

数组是将元素在内存中连续存储的；它的优点：因为数据是连续存储的，内存地址连续，所以在查找数据的时候效率比较高；它的缺点：在存储之前，我们需要申请一块连续的内存空间，并且在编译的时候就必须确定好它的空间的大小。在运行的时候空间的大小是无法随着你的需要进行增加和减少而改变的，当数据两比较大的时候，有可能会出现越界的情况，数据比较小的时候，又有可能会浪费掉内存空间。在改变数据个数时，增加、插入、删除数据效率比较低链表是动态申请内存空间，不需要像数组需要提前申请好内存的大小，链表只需在用的时候申请就可以，根据需要来动态申请或者删除内存空间，对于数据增加和删除以及插入比数组灵活。还有就是链表中数据在内存中可以在任意的位置，通过应用来关联数据（就是通过存在元素的指针来联系）

185. Java中ArrayList和Linkedlist区别?

ArrayList和Vector使用了数组的实现，可以认为ArrayList或者Vector封装了对内部数组的操作，比如向数组中添加，删除，插入新的元素或者数据的扩展和重定向。

LinkedList使用了循环双向链表数据结构。与基于数组的ArrayList相比，这是两种截然不同的实现技术，这也决定了它们将适用于完全不同的工作场景。

LinkedList链表由一系列表项连接而成。一个表项总是包含3个部分：元素内容，前驱表和后驱表，如图所示：

在下图展示了一个包含3个元素的LinkedList的各个表项间的连接关系。在JDK的实现中，无论LikedList是否为空，链表内部都有一个header表项，它既表示链表的开始，也表示链表的结尾。表项header的后驱表项便是链表中第一个元素，表项header的前驱表项便是链表中最后一个元素。

186. 请用两个队列模拟堆栈结构

两个队列模拟一个堆栈，队列是先进先出，而堆栈是先进后出。模拟如下队列a和b

（1）入栈：a队列为空，b为空。例：则将”a,b,c,d,e”需要入栈的元素先放a中，a进栈为”a,b,c,d,e”

（2）出栈：a队列目前的元素为”a,b,c,,d,e”。将a队列依次加入Arraylist集合a中。以倒序的方法，将a中的集合取出，放入b队列中，再将b队列出列。代码如下：

```
1. public static void main(String[]args){

2.       Queue<String>queue=new LinkedList<String>();//a队列

3.       Queue<String>queue2=new LinkedList<String>();    //b队列

4.       ArrayList<String>a=new ArrayList<String>();      //arrylist集合是中间参数

5.       //往a队列添加元素

6.       queue.offer("a");

7.       queue.offer("b");

8.       queue.offer("c");

9.       queue.offer("d");

10.      queue.offer("e");

11.              System.out.print("进栈：");

12.      //a队列依次加入list集合之中13.              for(Stringq:queue){

14.      a.add(q);
```

```
15.             System.out.print(q);
16.     }
17.     //以倒序的方法取出（a队列依次加入list集合）之中的值，加入b对列
18.     for(int i=a.size()-1;i>=0;i--){
19.             queue2.offer(a.get(i));
20.     }
21.     //打印出栈队列
22.     System.out.println("");
23.     System.out.print("出栈：");
24.     for(String q:queue2){
25.             System.out.print(q);
26.     }
27. }
```

打印结果为（遵循栈模式先进后出）：

进栈：abcde

出栈：edcba

187. Collection和Map的集成体系

```
Collection：

Map：
```

188. 什么是线程池，如何使用?

线程池就是事先将多个线程对象放到一个容器中，当使用的时候就不用new线程而是直接去池中拿线程即可，节省了开辟子线程的时间，提高的代码执行效率。

在JDK的java.util.concurrent.Executors中提供了生成多种线程池的静态方法。

```
1.ExecutorService newCachedThreadPool=Executors.newCachedThreadPool();
2.ExecutorService newFixedThreadPool=Executors.newFixedThreadPool(4);
3.ScheduledExecutorService new ScheduledThreadPool=Executors.newScheduledThreadPool(4);
4.ExecutorService newSingleThreadExecutor=Executors.newSingleThreadExecutor();
```

然后调用他们的execute方法即可。

189. 常用的线程池有哪些?

newSingleThreadExecutor：创建一个单线程的线程池，此线程池保证所有任务的执行顺序按照任务的提交顺序执行。

newFixedThreadPool：创建固定大小的线程池，每次提交一个任务就创建一个线程，直到线程达到线程池的最大大小。

newCachedThreadPool：创建一个可缓存的线程池，此线程池不会对线程池大小做限制，线程池大小完全依赖于操作系统（或者说JVM）能够创建的最大线程大小。

newScheduledThreadPool：创建一个大小无限的线程池，此线程池支持定时以及周期性执行任务的需求。

newSingleThreadExecutor：创建一个单线程的线程池。此线程池支持定时以及周期性执行任务的需求。

190. 请叙述一下您对线程池的理解?

（如果问到了这样的问题，可以展开的说一下线程池如何用、线程池的好处、线程池的启动策略）合理利用线程池能够带来三个好处。

第一：降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。第二：提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。

第三：提高线程的可管理性。线程是稀缺资源，如果无限制的创建，不仅会消耗系统资源，还会降低系统的稳定性，使用线程池可以进行统一的分配，调优和监控。

191. 说说你对Java中反射的理解

Java中的反射首先是能够获取到Java中要反射类的字节码，获取字节码有三种方法，1.Class.forName(className)

2. 类名.class3.this.getClass()。然后将字节码中的方法，变量，构造函数等映射成相应的Method、Filed、Constructor等类，这些类提供了丰富的方法可以被我们所使用。

192. 动静态代理的区别,什么场景使用?

静态代理通常只代理一个类，动态代理是代理一个接口下的多个实现类。

静态代理事先知道要代理的是什么，而动态代理不知道要代理什么东西，只有在运行时才知道。

动态代理是实现JDK里的InvocationHandler接口的invoke方法，但注意的是代理的是接口，也就是你的业务类必须要实现接口，通过Proxy里的newProxyInstance得到代理对象。

还有一种动态代理CGLIB，代理的是类，不需要业务类继承接口，通过派生的子类来实现代理。通过在运行时，动态修改字节码达到修改类的目的。

AOP编程就是基于动态代理实现的，比如著名的Spring框架、Hibernate框架等等都是动态代理的使用例

193.你所知道的设计模式有哪些

Java中一般认为有23种设计模式，我们不需要所有的都会，但是其中常用的几种设计模式应该去掌握。下面列出了所有的设计模式。需要掌握的设计模式我单独列出来了，当然能掌握的越多越好。

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。