Java 基础篇

1. String 类中常用的方法?

split()：把字符串分割成字符串数组
indexOf()：从指定字符提取索引位置
append()：追加字符或字符串
trim()：去除字符串两端空格
replace()：替换
hashCode()：返回此字符串的哈希码
subString()：截取字符串
equals()：比较
length()：获取字符串的长度
valueOf()：转换为字符串
concat()：将指定字符串连接到此字符串的结尾
compareTo()：用来比较两个字符串的字典顺序
compareToIgnoreCase()：不考虑大小写,按字典顺序比较两个字符串
contains()：　检查一个字符串中是否包含想要查找的值
用 string 类里的哪一个方法替换掉第五个元素?

```
String str = new String("helloWorld");

System.out.println(str.replace(5, 'T'));
```

2. 什么是重载和重写？

我理解得 Java 里方法的重写(Overriding)和重载(Overloading)就是指的 java 里多态，重写就是父类与子类之间的多态，重载是在一个类中多态对的体现　(1)方法重载是一个类中定义了多个方法名相同,而他们的参数的数量不同或数量相同而类型和次序不同,则称为方法的重载(Overloading)。
　(2)方法重写是在子类存在方法与父类的方法的名字相同,而且参数的个数与类型一样,返回值也一样的方法,就称为重写(Overriding)。

3. String 中的==和 equals 的区别?

"=="比较的是两个字符串的内存地址。 "equals"比较的是两个字符串的实际值。

== 对于简单类型是值比较，对于对象类型是地址比较

equals 是方法，通常是比较两个对象的 hashCode，对于字符串，hashCode 是通过字符串的内容算出来的，所以间接可以比较两个的内容

4. Java 中的 String，StringBuilder，StringBuffer 三者的区别?

在开发中 String 我们主要用于少量的字符串操作，他修饰的都是不可变字符串. StringBuffer :我们主要用于可变的字符串中,我们在拼接 Sql 语句的时候用的,还有就是他是线程安全的，在字符串后面拼接的方法是 append. Stringbuilder 这个我开发的时候没用过,也没了解过

5.Final 关键字

其实 Final 是一个安全修饰符,就是用 final 修饰的类不能被继承,用 final 声明的方法不能被重写,使用 final 声明的变量就相当于常量,不能被修改。

6.final 和 finally 的区别?

Finally 是在异常里经常用到的，就是 try 和 cach 里的代码执行完以后,必须要执行的方法,我们经常在 finally 里写一些关闭资源的方法,比如说关闭数据库连接,或者关闭 IO 流什么的.

7.Java 里可不可以有多继承？

不可以，想多继承的话怎么办？ 用接口

8.HashMap 和 Hashtable 的区别

这两个都是是 Map 接口下的实现类，我们开发的时候经常用的是 HashMap 虽然 HashMap 不是线程安全的但是他的存储效率比较高,Hashtable 是线程安全我们看过他底层的 put 方法前面加的 synchronized 关键字,但是他的效率太低,我们开发很少用.**为什么说 hashTable 是线程安全的？** 因为看源码的时候里面的 put 方法前面加了 synchronized 关键字,**什么情况下用 hashMap？** 我们有时候在自定义返回 json 字符串的时候,封装到 List 中进行返回.**你知道 Hashmap 的底层实现吗？** 参考后面的答案

Map 集合有哪些实现类，分别具有什么特征

HashMap 线程不安全的键值对集合 允许 null 值，key 和 value 都可以
HashTable 线程安全的键值对集合 不允许 null 值，key 和 value 都不可以（不用）
TreeMap 能够把它保存的记录根据键排序,默认是按升序排序，也可以指定排序的比较器（基本不用）

如何使用 Map 集合来保证线程安全

HashTable 效率低
Collections.synchronizedMap 方法将 HashMap 转换为安全的 效率低
使用 java.util.Map 下的 ConcurrentHashMap

9.解决 hashmap 线程不安全问题？

可以通过 collections 集合工具类对不安全的线程进行包装，使其变成线程安全的，也可以在使用其时加 synchronized 关键字进行同步

10.Hashmap 的底层实现原理

这个我在论坛上看过,Hashmap 底层是通过数组和链接联合实现的，当我们创建 hashmap 时会先创建一个数组，当我们用 put 方法存数据时，先根据 key 的 hashcode 值计算出 hash 值，然后用这个哈希值确定在数组中的位置，再把 value 值放进去，如果这个位置本来没放东西，就会直接放进去，如果之前就有，就会生成一个链表，把新放入的值放在头部，当用 get 方法取值时，会先根据 key 的 hashcode 值计算出 hash 值，确定位置，再根据 equals 方法从该位置上的链表中取出该 value 值。
参考文章
https://my.oschina.net/u/3954808/blog/3051948

11.hash 碰撞怎么产生，怎么解决？

这道题我正好在上家面试的时候问到过,当时我也不会,我回去专门上网查了一下,解决方案,堆里面在存储对象地址的时候有一个开放地址的方法：当发生地址冲突时，按照某种方法继续探测哈希表中的其他存储单元，直到找到空位置为止。
第二种是用 rehash(再哈希法)：当发生冲突时，使用第二个、第三个、哈希函数计算地址，直到无冲突时。缺点：计算时间增加。比如上面第一次按照姓首字母进行哈希，如果产生冲突可以按照姓字母首字母第二位进行哈希，再冲突，第三位，直到不冲突为止
3、链地址法（拉链法）：创建一个链表数组，数组中每一格就是一个链表。若遇到哈希冲突，则将冲突的值加到链表中即可。（java 里的 hashMap 就是使用的这种方法）

12.int 和 Integer 的区别？

Int 是基本数据类型呀 Integer 是 java 为 int 提供的封装类，是引用数据类型；Int 的默认值为 0，而 integer 的默认值为 null

13.Java 的三大特性

这个 java 的三大特性无非就是封装，继承，多态嘛
封装体现到我们把方法和变量都封装到一个类里,直接用对象调用就可以了
继承的体现是我们可以复用父类里的代码,我们在项目中可以把公有的方法写到 base 的一个类中,让其他类继承就可以了.
多态就是 Java 里的继承，重写，重载,都是多态,如果两边是数字的化,那就是一个运算符,如果两边是字符串的话,那他就是个连接符.

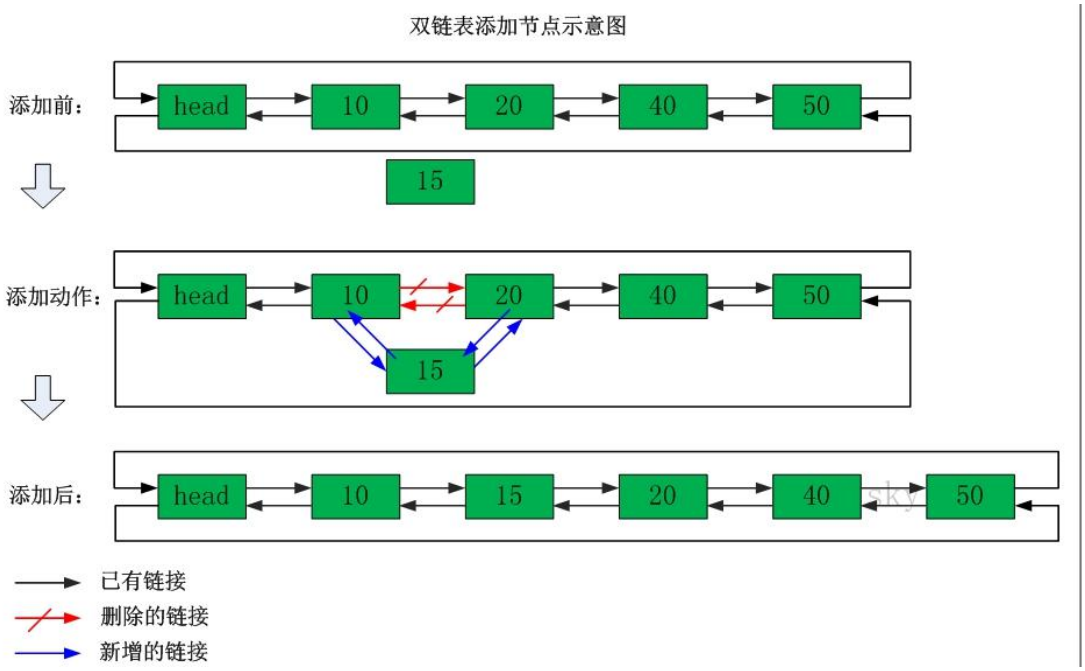
14.java arrayList 的存储结构,初始化的时候创建多大的数组?

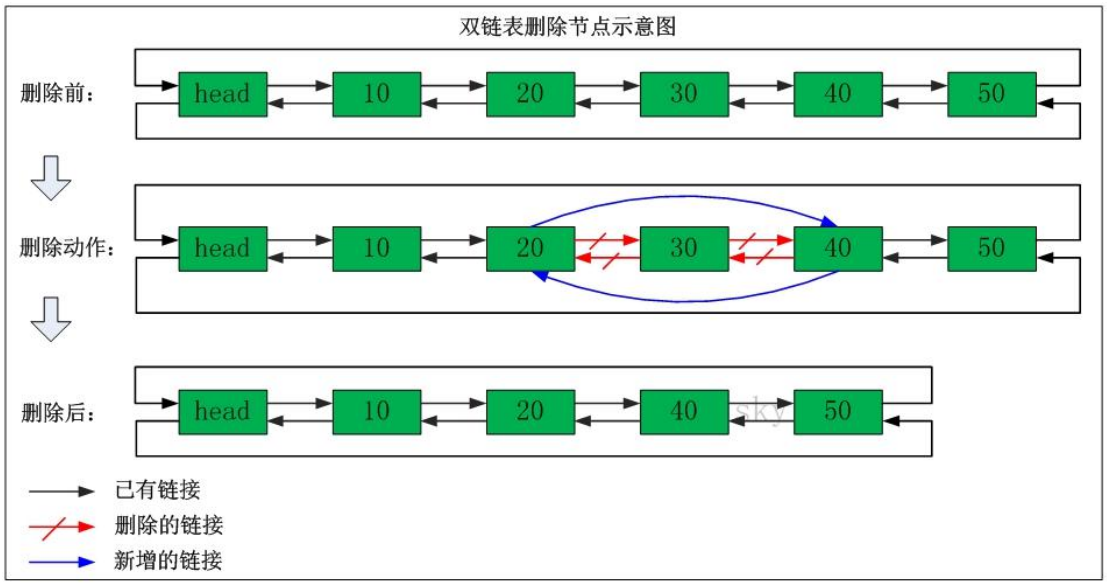
ArrayList 是基于数组实现的，是一个动态数组，其容量能自动增长，类似于 C 语言中的动态申请内存，动态增长内存。
ArrayList 是线程不安全的，只能用在单线程环境下，多线程环境下可以考虑用 Collections.synchronizedList(List l)函数返回一个线程安全的 ArrayList 类，也可以使用 concurrent 并发包下的 CopyOnWriteArrayList 类。

15.ArrayList 与 LinkedList 区别

ArrayList 使用数组方式存储数据，所以根据索引查询数据速度快，而新增或者删除元素时需要设计到位移操作，所以比较慢。
LinkedList 使用双向链接方式存储数据，每个元素都记录前后元素的指针，所以插入、删除数据时只是更改前后元素的指针指向即可，速度非常快，然后通

过下标查询元素时需要从头开始索引，所以比较慢，但是如果查询前几个元素或后几个元素速度比较快。开发中什么时候到 ArrayList?,我们在做查询的时候把查询出来的数据经常存到 arraylist 里.





2.1 List 集合有哪些实现类，分别具有什么特征

常见的 List 实现类
ArrayList 数组实现 查询快 增删慢
LinkedList 链表实现 查询码 增删快 项目中基本不用 大部分是查询
Vector 数组实现 线程安全 效率低 基本不用

2.2 如何使用的 List 集合来保证线程安全

1.使用 Vector

2.使用 Collections 中的方法 synchronizedList 将 ArrayList 转换为线程安全的 List

3.使用 java.util.current 包下的 CopyOnWriteArrayList（推荐）

16.抽象类和接口的区别?

抽象类里可以有普通方法也可以有抽象方法,接口里只能有抽象方法,但是在jdk1.8以后也可以在接口中添加普通方法了,就是得用default修饰一下这个方法,调用方式如下代码:这个代码不用背,在于理解和动手操作

```
interface Vehicle {
    default void print(){
        System.out.println("我是一辆车!");
    }
    static void blowHorn(){
        System.out.println("按喇叭!!!");
    }
}
必须是default (默认)或者 static (静态)修饰的方法，这样，接口中也可以写实现，
而且接口的实现类可以重新接口的默认方法，也可以用 super 关键字调用接口的默认方法!
public class car implements vehicle, fourWheeler {
    default void print(){
        System.out.println("我是一辆四轮汽车!");
    }
}
public class car implements vehicle, fourWheeler {
    public void print(){
        vehicle.super.print();
    }
}
```

17.什么是接口?有没有做过接口开发?

你是说的前后端交互的接口是吧,我主要就是做Java后台开发的,我们开发的时候有一个接口文档,我们按照接口文档里规定的返回类型参数请求的URL进行开发就行,我们一般用postman做的测试,接口文档里主要包括什么?请求的URL，参数，返回值类型，请求方式get，post,put,delete,以下接口文档不用背,只是让看下知道什么是接口文档,有印象就行

请求方法	GET
URL	http://sso.taotao.com/user/check/{param}/{type}
参数说明	格式如：zhangsan/ 1，其中 zhangsan 是校验的数据，type 为类型，可选参数 1、2、3 分别代表 username、phone、email 可选参数 callback：如果有此参数表示此方法为 jsonp 请求，需要支持 jsonp。
示例	http://sso.taotao.com/user/check/zhangsan/1

返回值	{
	status: 200 //200 成功
	msg: "OK" // 返回信息消息
	data: false // 返回数据，true：数据可用，false：数据不可用
	}

18.泛型

我记得泛型是 JDK1.5 以后有的,我觉得使用泛型的好处就是让我们的集合变的更安全一些,省去了强制转换.

19.IO 和 NIO 的区别?

这个 NIO 是 JDK1.7 以后有的,它们俩的主要区别是:io 是面向流是阻塞 io,nio 是面向缓冲,非阻塞的 io;
io 的话每次从流中读取一个或多个字节,直到读取完所有的字节,没有缓存到任何地方.nio 读取的是数据是有缓存,就是说他读取的数据是在缓冲里读的.
另外的话,java 中的各种 io 是阻塞的.就是说一个线程调用 read 或者 write()时,这个线程就已经被阻塞了,直到读取到一些数据为止,或者是完全写入.在这个过程中不能干其他的事情. nio 的非阻塞模式,当发送一个读取数据的请求的时候,如果没有读取到可用的数据,就什么也不会获取,且不会让线程阻塞写也是这样.非阻塞的 IO 的空闲时间可用用来做其他的操作所以,一个单独的非阻塞线程可以管理多个输入和输出通道,另外 NIO 还有一个 selector(选择器),它是可以管理多个输入输出的通道.我大概了解的也就是这样.

20. 静态变量与非静态变量的区别如下?

我们在开发的时候尽量是少使用或者不适用静态的变量我们会尽量把常量写到 properties 配置文件里,直接从配置文件中读取. 因为使用静态变量以后在程序初始化的时候就会存到内存中而且直到应用程序结束,应用程序就是 tomcat 启动(红色字体不用背在于理解),太占用 jvm 的内存,使用非静态的化,被实例化以后才会给分配内存, 后面的概念问到的说,不问就不说了(说多了感觉像是背的.)

调用方式

静态变量只能通过“类.静态变量名”调用，类的实例不能调用；
非静态变量当该变量所在的类被实例化后，可通过实例化的类名直接访问。

共享方式

静态变量是全局变量，被所有类的实例对象共享，即一个实例的改变了静态变量的值，其他同类的实例读到的就是变化后的值；
非静态变量是局部变量，不共享的。

访问方式

静态成员不能访问非静态成员；

非静态成员可以访问静态成员。

21.在 Java 中要想实现多线程代码有三种手段?

一种是继承 Thread 类
另一种就是实现 Runnable 接口
最后一种就是实现 Callable 接口
(第四种也是实现 callable 接口，只不过有返回值而已)

22.线程的状态

其实线程一般具有五种状态，即创建、就绪、运行、阻塞、终止。

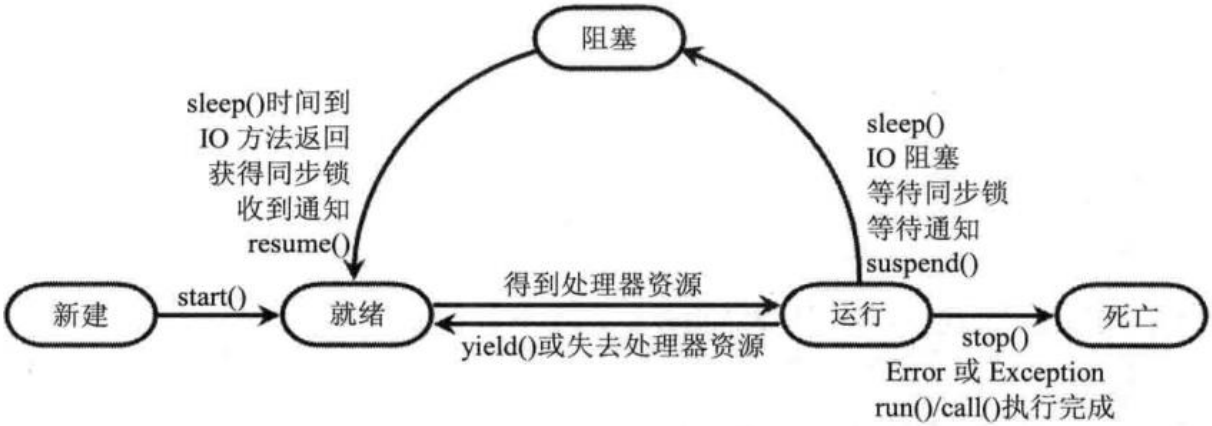


图 16.4　线程状态转换图

下面这些问到的就回答一下,比如说让你启动一个线程调用哪个方法?
new 出一个 Thread 对象后，些时线程是新建
start()之后线程是就绪状态，等待 CPU 调用 CPU 调用之后线程就变成了运行：
sleep 之后是阻塞，不会释放监视器上的锁，线程睡起来之后又变成就绪状态
yield 之后就绪状态，不会释放锁
wait 之后，其它线程变成阻塞状态
nifity 之后，其它线程变成就绪状态
suspend：就是将线程挂起变成阻塞状态
resume：就是挂起的线程恢复，变成就绪状态
stop：停止线程，变成死亡状态
run 方法运行完或出现异常或 break 等变成死亡状态

23.线程同步,什么是锁?

实现线程同步有三种方式：

1) **同步代码块**：在代码块上加上“synchronized”关键字的话，则此代码块就称为同步代码块。

同步代码块格式：

```
synchronized(监视对象){
        需要同步的代码 ；
    }
```

解释：监视对象有三种：对象、String、.class 文件（只要是不变的对象都可以做监视对象）

2） **同步方法**

同步方法定义格式：

```
synchronized 方法返回值 方法名称(参数列表){}
```

在方法上加 synchronized，是把当前对象做为监视器

3） **同步锁**

```
Lock lock = new  ReentrantLock();(可以在类中直接 new)
lock.lock();  中间的代码块进行加锁 lock.unlock();
```

24.线程通信

- 传统的线程通信－wait()与 notify()方法
- notify()和 notifyAll()

对于唤醒的操作有两个：notify()、notifyAll()。一般来说，所有等待的线程会按照顺序进行排列，如果现在使用了 notify()方法的话，则会唤醒第一个等待的线程执行，而如果使用了 notifyAll()方法，则会唤醒所有的等待线程，那个线程的优先级高，那个线程就有可能先执行。

参考文章:https://blog.csdn.net/pengzhisen123/article/details/79455742

25.多线程怎么解决高并发？

synchronized 关键字主要解决多线程共享数据同步问题。

ThreadLocal 使用场合主要解决多线程中数据因并发产生不一致问题。

ThreadLocal 和 Synchonized 都用于解决多线程并发访问但是 ThreadLocal 与 synchronized 有本质的区别:

synchronized 是利用锁的机制，使变量或代码块在某一时该只能被一个线程访问。而 ThreadLocal 是为每一个线程都提供了变量的副本，使得每个线程在某一时间访问到的并不是同一个对象，这样就隔离了多个线程对数据的数据共享。而 Synchronized 却正好相反，它用于在多个线程间通信时能够获得数据共享。

26.什么是多线程?在哪里应用过?

线程就是可执行的代码段，线程要服务于进程，一个进程有好多个线程，main方法就是主线程，多个线程同时执行就是多线程，线程池就是把准备好的线程放到线程池里，如果处理请求需要调用线程的时候，就从线程池里去调用，用完以后再放回到线程池里，这样就防止高并发节省资源,我们目前开发中没有涉及到过多线程,其实我个人理解用多线程无非是为了

提高代码的执行效率提高客户的体验,解决高并发,但是项目里如果多线程使用的多的话,后期的代码维护这一块也不怎么好维护,我们现在在解决这些高并发都是建议使用中间件来解决,redis啦,activeMQ啦,还有solr等等, 然后把这几个中间件的原理说一下....就能巧妙的避免开这个多线程的问题啦...

Java 中的锁有几种方式

两种:
Synchronized
Lock

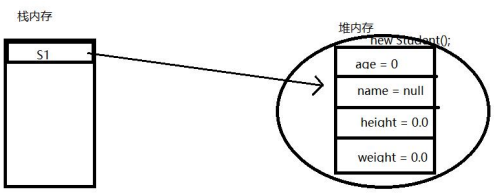
Synchronized的局限性:
1.如果这个获取锁的线程由于要等待IO或者其他原因（比如调用sleep方法）被阻塞了，但是又没有释放锁，其他线程便只能干巴巴地等待。(不能主动释放锁)
2.当有多个线程读写文件时，读操作和写操作会发生冲突现象，写操作和写操作会发生冲突现象，但是读操作和读操作不会发生冲突现象如果多个线程都只是进行读操作，所以当一个线程在进行读操作时，其他线程只能等待无法进行读操作。（不分情况，一律锁死）

Lock 的几个实现类

ReentrantLock是一个可重入的互斥锁，又被称为“独占锁”
ReadWriteLock，顾名思义，是读写锁。它维护了一对相关的锁——“读取锁”和“写入锁”，一个用于读取操作，另一个用于写入操作。他的两个实现类读锁readerLock和写锁writerLock。

27.jvm 内存分析？解释下堆和栈？解释下虚拟机?

我原来学java的时候知道JVM内存结构主要有三大块：堆内存、方法区和栈。堆内存是JVM中最大的一块内存地址,它主要由年轻代和老年代还有持久代组成,所有new出来的对象都存储在该区域. 栈就是暂存数据的地方,每个线程包含一个栈区，栈存放在一级缓存中，存取速度较快,栈中只保存基础数据类型的对象和自定义对象的引用.每个栈中的数据都是私有的，其他栈不能访问。方法区存放了要加载的类的信息（如类名、修饰符等）、静态变量、构造函数、final定义的常量、类中的字段和方法等信息。
声明并创建对象
1. 访问属性



28. 自定义 Annotation,自定义注解

大学里学 java 初级的时候写过自定的注解， 按照下面代码说一下，开发以后就没在写过了，都是用的框架给带的注解 spring 的那些注解给挨个说一遍

向 Annotation 中设置参数内容

Annotation 定义格式：

```
[public] @interface Annotation 名称{
        数据类型 变量名称();
    }
```

定义注解：

```
public @interface Controller {

}
```

使用注解：

```
@Controller
public class UserController {

}
```

定义带参数的注解（详见 api，有时间可以写个 demo）

定义注解：

```
public @interface Controller {
    public String value();          // 定义参数
}
```

使用注解：

```
//@Controller(value="userService")
@Controller("userService")   // 注解的参数名为 value 时，参数名可以省略
public class UserController {
}
```

常见 java 内建注解，override 等

29.有没有遇到过内存溢出，内存泄漏?

开发的时候遇到过,其实内存溢出指的是 jvm 内存溢出，通过调整堆，栈的大小来解决。代码中出现死循环或递归调用也会造成内存溢出,一般我们去调整一下他的内存大小就行,配置一下 tomcat 里的一个参数就能调整,一般造成内存溢出的原因是什么？一方面有可能是代码中有长时间没有关闭的连接,或者是用 IO 读取完文件以后没有及时关闭,原来再使用 hibernate 的时候就遇到过内存溢出,就是因为没有及时的去关闭缓存造成的,还有就是代码问题,比如说写了死循环,程序出问题,递归自己调用自己的时候代码写的有问题都有可能造成内存溢出.我感觉内存泄漏和内存溢出差不多.

30.说一下 GC 垃圾回收过程

我们 Java 中的垃圾回收都是自动的,我们很少几乎不去手动的干预,我就是先说一下这个 JVM 中的的堆内存结构,他主要分为新生代和老年代，新生代就是用来存放刚被 new 出来的对象，一般情况下占堆的 1/3 空间,还有就是新生代中又分为 3 个区具体我也记不清是哪三个区了 反正老年代里存放咱们整个应用程序中生命周期长的内存对象,我原来再网上看过,我们的 JVM 垃圾回收算法我记得差不多有 4 种,标记清除法,复制算法,标记整理算法,分代收集算法,我就了解其中的两种,比如说 说清楚其中的两个就行,都说出来就有点太假了,根本不可能记不住那么多

第一种就是标记-清除法

分为：标记，请除

标记阶段：直接在内存里标记无用的对象，然后清除阶段直接回收被标记的对象；

缺点：形成内存碎片,一些大的对象无法找到足够的空间而触发新的垃圾收集动作。

第二种：复制算法：

将内存划分为大小相等的两块，当一块的内存用完了，就把还存活的对象复制到另外一块上面，然后将之前的那块清理掉

缺点：浪费内存太多（对老年代的使用，效率低）

第三种：标记-整理算法

将存活的对象都向一端移动，然后直接清理掉这端边界以外的内存

第四种：分代收集算法：（当前商业虚拟机都采用这个）

根据对象的存活中期的不同将内存划分为几块，一般 Java 堆分为新生代和老年代

新生代：用复制算法　老年代用标记整理算法进行回收

31.你以前有遍历过 xml 文件吗?

我们使用 Dom4J 遍历 xml 文件,我们在遍历的时候先获取根节点再获取子节点,或者里面属性啥的.getRoot,getDocument 方法啥的,

32.什么是反射，反射能干嘛?

反射：简单说,反射机制值得是程序在运行时能够获取自身的信息。在 java 中,只要给定类的名字，那么就可以通过反射机制来获得类的所有信息。反射的作用其实就是:在运行时能够判断任意一个对象所属的类，还有就是在运行时构造任意一个类的对象,我们常用的 Spring 框架也是利用 Java 反射这一块架构的还有就是 在运行时判断任意一个类所具有的成员变量和方法,还能在在运行时调用任一对象的方法,还有在运行时创建新类对象

看下面的代码是在于理解什么是反射,不用背出来

```
public class Base {
    static int num = 1;
    static {
        System.out.println("Base " + num);
    }
```

```
}public class Main {
    public static void main(String[] args) {
        // 不会初始化静态块
        Class clazz1 = Base.class;
        System.out.println("------");
        // 会初始化
        Class clazz2 = Class.forName("zzz.Base");
    }
}
```

33.get 和 post 请求的区别?

Get 是用 url 传递数据的不安全,而且传递的数据量比较小,post 是可以传递大数据量,其实我们还有在接口开发里还有 put 请求修改, delete 请求方式啥的.

34.说五个开发中常见的异常?

我们开发经常会遇到比如说 IOException, NullpointException, FileNotException,SQLException,ClassNotFoundException,ClassCastException, pathException,还有公司里有时候会有自己的异常框架,直接继承

35. Throw 和 Throws 的区别?

throw 是抛出自定义异常的,throws 是在方法上抛出异常的.

36.session 和 cookie 区别

Session 是保存到服务器端,cookie 是保存到客户端的,session 相对于 cookie 来说更安全一些,如何从 cookie 中取数据? request.getCookie,session 数据的共享问题如何解决,参考后面答案.

37.日期转换帮助类常用的方法?

```
String sDate = "";
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        Date dt = getDate(date);
        sDate = sdf.format(dt);
```

38.HTTP 和 HTTPS 的区别

我们公司用的是HTTPS协议 他俩一个是加密 一个不加密 HTTPS是基于http开发的是http的安全版 HTTPS协议需要到ca申请证书 一般免费证书很少 需要交费 他们两个链接的端口引入不一样 http是80 https是443http是超文本传输协议 信息是明文传输的HTTPS则是具有安全性的ssl加密传输协议

39.jdk1.8 和 1.7 区别

我了解JDK8的新特性有:就是接口里也可添加普通的方法了,不用非得是抽象方法了,但是必须得用default进行修饰,调用的时候也还是得实现这个接口以后才能调用,还有就是添加了一个Lambda 表达式,让我们遍历集合数据的时候速度更快,我大概就知道这些,我们目前开发也还没用到过这些新特性,对了,还有就是JDK1.8以后的map存储方式是1.7的也有些区别.如果不相等则形成链表结构， jdk1.7后加的在前面，先加的移下，这种情况叫碰撞。这种碰撞的情况应尽量避免，否存一个索引中链表的数据大量时，该索引当再次插入一个对象时equals比较全部影响效率。

因此jdk1.8改善这种碰撞情况的出现，jdk1.8中的HashMap存储结构是由数组、链表、红黑树这三种数据结构形成，红黑树查询删除快新增慢。

参考文章https://blog.csdn.net/changhangshi/article/details/82114727

40.TCP 协议,UDP 协议,RPC 协议,JMS 协议?

我知道这些都是通信协议tcp跟udp都是传输协议 主要区别是tcp协议连接需要3次握手断开需要四次挥手是通过流来传输的就是确定连接后一直发送信息传完后断开,Udp：udp不需要进行连接直接把信息封装成多个报文直接发送所以速度更快,Rpc:一种通过网络从远程计算机程序上请求服务 不需要了解底层网络技术的协议.Jms：是java的消息服务 jms的客户端之间通过jms服务进行异步的消息传输,我以前没有做过socket编程和网络编程这一块,公司用的化可以学

41.Jvm 优化?

其实优化这一块我也说不太好,我就照我理解的方式说一下,比如说我们得遵守咱们 Java 的编码规范,比如说尽量少使用静态变量,循环最好是不要超过三层以上的嵌套,因为后期维护起来比较麻烦,也比较耗费 jvm 的内存,去掉项目中不必要的 jar 包,如果感觉 jvm 内存不够用,如果你的应用是跑到 tomcat 服务器上的话可以去修改下 catalina.bat 文件设置下 Xms 初始化堆的大小,和 XXm 最大允许分配堆内存,其实这个也是不设置的越大越好,也得经常更改,在我个人理解上如果设置的越大,有可能垃圾回收机制不会及时回收不长用的对象.

42.Js 中的基本数据类型?

1. Undefined:只有一个值，就是 undefined
2. Null;只有一个值，就是 null,逻辑角度看，null 值表示一个空对象指针
3. Boolean:类型只有两个值：true 和 false
4. Number:使用 IEEE754 格式来表示整数和浮点数值
5. String:由单引号或双引号括起来的字符序列,任何字符串的长度都可以通过访问 length 属性获得

43.js 中的==和===的区别?

”==”与”===”是不同的,前者是判断值是否相等,后者是判断值及类型是否完全相等。

44.转发和重定向的区别

这个我知道,重定向就是 RequestDispatcher.forward 方法只能将请求转发给同一个项目中,而且可以携带数据；而 HttpServletResponse.sendRedirect 方法可以跳转到当前项目中的页面也可以跳转到其他项目中的页面,但是他不能携带数据

45.Collection 下的接口用过哪些?

知道这个接口下有两个 List 和 set 子接口.

46 类的初始化过程

Student s = new Student();在内存中做了哪些事情?

2. 加载 Student.class 文件进内存
3. 在栈内存为 s 开辟空间
4. 在堆内存为学生对象开辟空间
5. 对学生对象的成员变量进行默认初始化
6. 对学生对象的成员变量进行显示初始化
7. 通过构造方法对学生对象的成员变量赋值
8. 学生对象初始化完毕，把对象地址赋值给 s 变量

47.如何遍历 Map 集合?

使用 keyset 遍历.

```
Map<String,Object>  specMap = JSON.parseObject(item.getSpec());
for (String key:specMap.keySet()){
   title += specMap.get(key);
```

```
}
```

48.Java 中的各种锁?看下网址理解下

打开链接看下 java 中的各种锁
https://blog.csdn.net/loveqishan/article/details/88944852