# 第一部分：Java 语言核心

## 为什么创建的varchar类型的字段超出后自动转换为longtext类型

【关键字：varchar】【难度：初级】【掌握程度：了解】

解决方案：字段列类型都有自动转换

## 数据库不能停机，请问如何备份? 如何进行全备份和增量备份?

【关键字：数据备份】【难度：中级】【掌握程度：掌握】

解决方案：可以使用逻辑备份和双机热备份。

完全备份：完整备份一般一段时间进行一次，且在网站访问量最小的时候，这样常借助批处理文件定时备份。主要是写一个批处理文件在里面写上处理程序的绝对路径然后把要处理的东西写在后面，即完全备份数据库。

增量备份：对ddl和dml语句进行二进制备份。且5.0无法增量备份，5.1后可以。如果要实现增量备份需要在my.ini文件中配置备份路径即可，重启mysql服务器，增量备份就启动了。

## 查询mysql数据库中用户，密码，权限的命令

【关键字：MySQL权限】【难度：初级】【掌握程度：了解】

解决方案：

查看MYSQL数据库中所有用户

mysql> select distinct user,host from mysql.user;



查看数据库中具体某个用户的权限

mysql> show grants for 'root'@'%';

## 数据库乱码

【关键字：数据库乱码】【难度：初级】【掌握程度：死背】

解决方案：插入中文前，设置编码为utf8或gbk即可

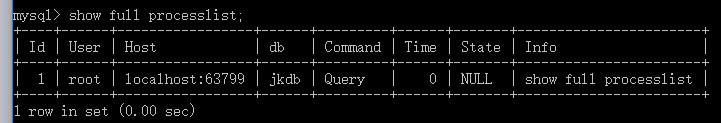
mysql> set names utf8;

mysql> set names gbk;

## 查看连接mysql的当前用户

【关键字：当前用户】【难度：初级】【掌握程度：掌握】

解决方案：show full processlist，在user字段中查看有哪些用户



## mysql服务启动失败

【关键字：MySQL启动服务报错】【难度：初级】【掌握程度：了解】

解决方案：通过百度用各种方式去解决，但是都不能解决，最后是通过重装mysql的方式来解决

## 输入命令报错

【关键字：命令报错】【难度：初级】【掌握程度：了解】

解决方案：忘记输入分号；

## mysql函数怎么理解

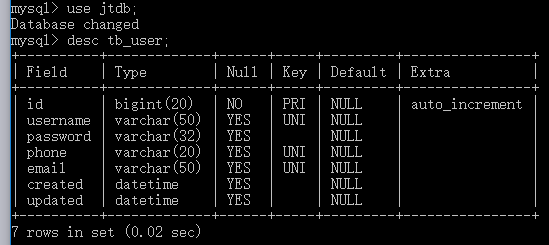
【关键字：MySQL函数】【难度：中级】【掌握程度：熟练】

解决方案：函数有很多，尽量记、记不住就记住几个常用的，其他可以通过查资料

## desc 表结构 int类型字段，显示int(11) 11是什么意思

【关键字：数据类型的长度】【难度：中级】【掌握程度：了解】

解决方案：11表示数字的个数 int最大值21亿 占用了10位数字，还有1个用来表示符号位



## 复制表数据的时候 批量插入数据的时候 自增主键 出现不连续的情况

【关键字：主键不连续】【难度：初级】【掌握程度：了解】

解决方案：和数据库引擎有关 MyISAM引擎是连续的 ，产生的原因可以参考http://blog.csdn.net/u014430366/article/details/70267601?utm\_source=itdadao&utm\_medium=referral 以及官方文档

https://dev.mysql.com/doc/refman/5.7/en/innodb-auto-increment-handling.html

## 外键设置是否越多越好

【关键字：外键】【难度：初级】【掌握程度：了解】

解决方案：外键体现了表之间的关系，但是外键也使得数据库的查询效率降低，所以外键应该尽量少

## 外键约束添加失败

【关键字：外键约束】【难度：初级】【掌握程度：了解】

解决方案：外键和参考的主键数据类型不一致。更改数据类型

## 第二次插入同样的数据失败

【关键字：重复插入数据】【难度：初级】【掌握程度：了解】

解决方案：主键没有设自增长，并且设置了默认值为0，第二次插入同样的数据违反了主键约束

## mysql中函数能否嵌套使用，例如max（min(salary)）

【关键字：嵌套使用函数】【难度：中级】【掌握程度：了解】

解决方案：sql语法中不能将函数嵌套使用，可以将一个函数的查询结果作为另外一个函数的查询条件

首先sql是有语法要求，不是随意写；再者一定要有业务逻辑，上面的页面逻辑首先不成立。

## 查询的SQL结果为空

查询平均工资最高的部门

select department\_id,avg(salary) g from employees group by department\_id

having g=(select max(avg) from (

select

department\_id,avg(salary) avg

from employees

group by department\_id) a);

【关键字：SQL正确执行后结果为空】 【难度：中级】【掌握程度：掌握】

解决方案：avg(salary)之后会出现小数，小数有精度丢失问题，=号左右两边的数据不相等就没结果集，查询的数据是正数的话是有结果集的。数据是小数的话 可以只保留2位小数 有结果集。Format(avg(salsry, 2))

## group by 后面两个字段什么意思

【关键字：group by】 【难度：初级】【掌握程度：了解】

解决方案：根据第一个字段分组，如果第二个分组字段相同将合并

## create table a(a double(15));为什么建表会报错

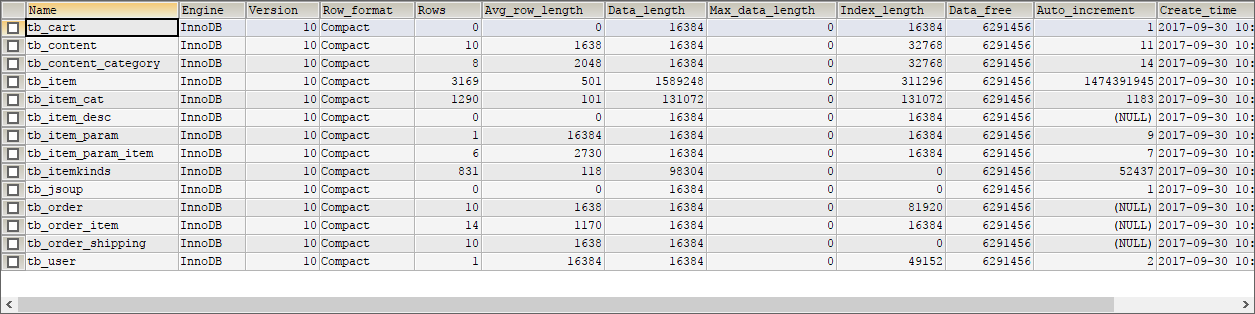
【关键字：create table】【难度：中级】【掌握程度：掌握】

解决方案：double类型有两个参数，一个是总长度，一个是小数点位数

## 查看所有表自增长的值

【关键字：查看自增长】【难度：中级】【掌握程度：掌握】

解决方案：使用show table stauts命令

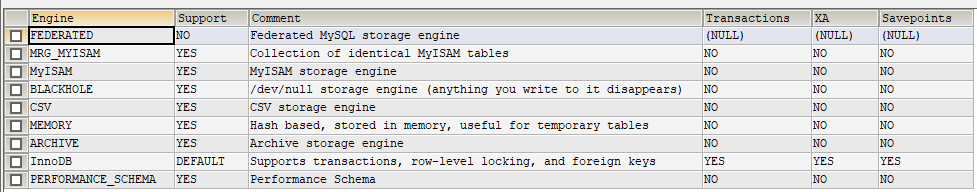


## 如何确定有哪些存储引擎可用

【关键字：数据库引擎】【难度：初级】【掌握程度：了解】

解决方案：mysql> show engines;

显示了可用的数据库引擎的全部名单以及在当前的数据库服务器中是否支持这些引擎



## MySQL怎么修复损坏的表

【关键字：MySQL表修复 几乎不用】【难度：专家】【掌握程度：了解】

解决方案：有两种方法，一种方法使用mysql的check table和repair table 的sql语句

CHECK TABLE tb\_user;

REPAIR TABLE tb\_user;

另一种方法是使用MySQL提供的多个myisamchk, isamchk数据检测恢复工具

## 怎么导出表结构

【关键字：导出表结构】【难度：初级】【掌握程度：掌握】

解决方案：

1.导出整个数据库

mysqldump -u用户名 -p密码 数据库名 > 导出的文件名

C:\Program Files\MySQL\MySQL Server 5.5\bin>mysqldump -uroot -proot jtdb > d:/jtdb.sql

2.导出一个表，包括表结构和数据

　　　　mysqldump -u用户名 -p 密码 数据库名 表名> 导出的文件名

　　　　C:\Users\jack> mysqldump -uroot -pmysql sva\_rec date\_rec\_drv> D:/date\_rec\_drv.sql

3.导出一个数据库结构

　　　　C:\Users\jack> mysqldump -uroot -pmysql -d sva\_rec > D:/sva\_rec.sql

4.导出一个表，只有表结构

　　　　mysqldump -u用户名 -p 密码 -d数据库名 表名> 导出的文件名

　　　　C:\Users\jack> mysqldump -uroot -pmysql -d sva\_rec date\_rec\_drv> D:/date\_rec\_drv.sql

5.导入数据库

　　　　常用source 命令

　　　　进入mysql数据库控制台，

　　　　如mysql -u root -p

　　　　mysql>use 数据库

　　　　然后使用source命令，后面参数为脚本文件(如这里用到的.sql)

　　　　mysql>source d:wcnc\_db.sql

## 导入sql文件多行显示未找到sql语句

【关键字：sql导入失败】【难度：初级】【掌握程度：了解】

解决方案：下载或者解压时出错，重新下载解压导入即可

## 控制台Mysql语句写完回车自动换行没反应，再输任何语句都没反应

【关键字：控制台回车后没反应】【难度：初级】【掌握程度：了解】

解决方案：可能输的语句中出现错误，ctrl+C退出重进

## 使用show variables like ' autocommit'; 查不到autocommit

【关键字：show variables】【难度：初级】【掌握程度：了解】

解决方案：学员使用的mysql版本是5.0，版本低的原因

## windows如何导入sql表

【关键字：导入SQL】【难度：初级】【掌握程度：掌握】

解决方案：使用source命令 比如：source D:\\test.sql

## windows系统下导入jtds数据库 mysql -uroot -p --default-character-set=字符集 jtds<路径，字符集u8没问题，gbk会出错

【关键字：导入乱码】【难度：初级】【掌握程度：掌握】

解决方案：用u8导入，查询的时候set character-set gb2312。

## 视图和索引的区别

【关键字：视图和索引】【难度：中级】【掌握程度：了解】

解决方案：从创建、删除、作用、意义等几方面进行说明

## 对事务的几种隔离级别分不清楚

【关键字：事务的隔离级别】【难度：高级】【掌握程度：掌握】

解决方案：针对老师的课上案例进行重点分析

## limit 10,10　两个10的含义分不清，不知道是页数还是条数

【关键字：limit】【难度：初级】【掌握程度：死背】

解决方案：从第11条开始，显示10条记录

## utf8\_bin跟utf8\_general\_ci的区别

【关键字：utf8\_bin跟utf8\_general\_ci的区别】【难度：初级】【掌握程度：了解】

解决方案：ci是 case insensitive, 即 "大小写不敏感", a 和 A 会在字符判断中会被当做一样的;

bin 是二进制, a 和 A 会别区别对待.

例如运行:

SELECT \* FROM table WHERE txt = 'a'

那么在utf8\_bin中你就找不到 txt = 'A' 的那一行, 而 utf8\_general\_ci 则可以.

## 模糊查询 \_ 需要转义 ，不是模糊查询的时候转义了查不到结果，不转义可以查到结果

【关键字：\_】【难度：初级】【掌握程度：死背】

解决方案：通配符只在like 查询时有作用 ，其他地方没作用 所以 不必转义

## 创建电商项目表的外键不能被引用

【关键字：外键引用】【难度：初级】【掌握程度：了解】

解决方案：被引用的外键必须具有唯一性约束

## mysql 事务隔离级别的区别

【关键字：事务的隔离级别】【难度：高级】【掌握程度：死背】

解决方案：分别解释 建表 用数据做测试 观察实验结果 可以参考：

http://www.jianshu.com/p/8d735db9c2c0

学员整体对事物隔离级别的认识很模糊

解决方案 在晚自习期给了他们合理的解释：级别就是Read uncommit Read commit Repeatable reasd Serializable这四种；脏读知识Read Uncommit该种级别才能进行

## MySql中any函数可以组合使用吗？例如=any，>any，<any等等，可以使用的话，那又在什么情况下使用？

【关键字：any】【难度：初级】【掌握程度：掌握】

解决方案：any函数相当于in 关键字，=any：函数中字句返回列可以是多列，

>any：函数中查询返回字段必须是单个字段。在比较条件是，需要用到多个字句字段的情况。

## MySql中查看表结构的时候，在字段后面有AK修饰，AK是什么意思？

【关键字：AK】【难度：初级】【掌握程度：了解】

解决方案：替代键的解释（alternate key）,替代键（alternate key）可以是数据表内不作为主键的其他任何列，只要该键对该数据表唯一即可。换句话说，在唯一列内不允许出现数据重复的现象。

## 为什么开两个终端，一个终端关闭自动提交SET autocommit=0; 另一个终端显示自动提交开启？

【关键字：autocommit】【难度：初级】【掌握程度：了解】

解决方案：重启mysql后就可以了

## 导入sql文件后乱码

【关键字：导入乱码】【难度：初级】【掌握程度：死背】

解决方案：修改字符集为gbk

## 调用存储过程插入多行数据id不连续

【关键字：存储过程id不连续】【难度：初级】【掌握程度：了解】

解决方案：告诉学生这个是mysql底层的问题，不影响，只要id不重复就行

## 调试maven的时候报错

【关键字：maven报错】【难度：中级】【掌握程度：熟练】

解决方案：检查是否关联settings配置文件

## MySql中存储过程中是否可以使用会话变量？

【关键字：MySQL中的会话变量】【难度：中级】【掌握程度：了解】

解决方案：””可以使用，具体为什么不清楚。

## 数据里面的return 是什么意思

【关键字：return】【难度：初级】【掌握程度：了解】

解决方案：Return 返回的是具体的值/值变量

## 创建外键的条件

【关键字：外键】【难度：初级】【掌握程度：了解】

解决方案：

1.两个表必须是InnoDB表，MyISAM表暂时不支持外键（据说以后的版本有可能支持，但至少目前不支持）；

2.外键列必须建立了索引，MySQL 4.1.2以后的版本在建立外键时会自动创建索引，但如果在较早的版本则需要显示建立；

3.外键关系的两个表的列必须是数据类型相似，也就是可以相互转换类型的列，比如int和tinyint可以，而int和char则不可以；

## 问题描述：表引擎设置成MyISAM 也可以创建出带外键的表

【关键字：数据库引擎】【难度：初级】【掌握程度：了解】

解决方案：创建语句是不会报错的，show create table 之后发现 创建的表并不具备外键

## 调用存储过程中出现问题，调用失败

【关键字：存储过程调用】【难度：初级】【掌握程度：了解】

解决方案：单词写错，修改以后就可以了

## 数据库所在磁盘空间已满。

【关键字：磁盘已满】【难度：初级】【掌握程度：了解】

解决方法：增加 MySQL 所在的磁盘空间或者清理一些无用文件

## 存储过程与函数的区别

【关键字：存储过程与函数的区别】【难度：初级】【掌握程度：了解】

解决方案：类比Java中的有返回值方法与无返回值方法进行解释说明

## a- - 操作在什么时候数值进行改变的

【关键字：a--】【难度：初级】【掌握程度：了解】

解决方案：在当前行操作完成之后,--操作执行,下次调用时数据改变

## double result = 1.0 - 0.9; 结果 0.09999999999999998 怎么提高精度，有哪些方法提高精度

【关键字：double精度】【难度：初级】【掌握程度：了解】

解决方案：

1. 用NumberFormat类来格式化计算结果，按照自己想要的结果进行格式化，缺点就是要手动去格式化，舍入方式不同结果不一定精确。

double result = 1.0 - 0.9;

NumberFormat nf = NumberFormat.getInstance();// 根据自己的需求格式化

String resultStr = nf.format(result);

2. 如果不介意自己记录十进制的小数点，而且数值不大，那么可以使用long ，int等基本类型，具体用int还是long要看涉及的数值范围大小，缺点是要自己处理十进制小数点，最明显的做法就是处理货币使用分来计算，而不用元（只涉及加减）。

int resultInt = 10 - 9;

double result = (double) resultInt / 10;//最终时候自己控制小数点

3. 使用BigDecimal来代替double，它能让你完全控制精度，结果会非常精确，加减乘除写起来也很方便，不过他有两个缺点：1，不是基本类型，与基本类型相比，操作起来不方便；2. 速度没有基本类型快，算是用速度换精度。相比第2个缺点并不要紧，但是第一个缺点会让你写起代码很不舒服。

String result = new BigDecimal("1").subtract(new BigDecimal("0.9")) .toString();

## 如何啥看表中的触发器

【关键字：触发器】【难度：初级】【掌握程度：了解】

解决方案 ：show triggers like '表名';

## char只有两位长度，中文字有8万多，怎么把超过65535的字赋值给char

【关键字：触发器】【难度：初级】【掌握程度：了解】

解决方案：java是用unicode编码的采用UTF-16，0x01---0x7F的字符编译后用单字节表示

0x00,0x80---0x7FF的字符编译后用双字节表示

0x800---0xFFFF的字符(如汉字)编译后用三字节表示。只收录了1W多个常见汉字

## eclipse 写完代码按工具栏绿色的运行按钮，控制台没任何结果

【关键字：eclipse】【难度：初级】【掌握程度：了解】

解决方案：直接按这个 运行的是上一次运行过的java文件，可以在当前类 右键选run as 运行当前这个java文件

## decimalformat四舍五入有的时候不进位？

【关键字：decimalformat】【难度：初级】【掌握程度：了解】

解决方案：Java语言设计是美国人的逻辑思维设计，应该是与国内的思维不是很相同，所以在四舍五入方面有这样一种说法：是这个函数的默认舍入策略是四舍六入五奇偶。具体策略如下。df.setRoundingMode(RoundingMode.HALF\_UP);

## Scanner获取控制台输入内容的时候抛出异常：InputMismatchException

【关键字：Scanner】【难度：初级】【掌握程度：了解】

解决方案：错误的解决方式就是输入的数据类型与接收的不匹配。

## double类型的数据为什么会有精度损失

【关键字：double】【难度：初级】【掌握程度：了解】

解决方案：该问题涉及到java底层保存double类型数据的数据结构，问题比较深入，可以给学员提供资料自己研究。

## 猜数字程序中第一次输入0会退出程序，但是先输入其他数字再次输入0不会退出程序

【关键字：猜数字】【难度：初级】【掌握程度：了解】

解决方案：应该将输入0的判断代码放到while循环中，讲师授课过程中也有这个问题

## Eclipse在使用过程中有警告提醒,如何修改

【关键字：eclipse】【难度：初级】【掌握程度：了解】

解决方案：保持良好的操作习惯,对于警告框中的内容理解,下次尽量避免

## 迭代，循环，遍历，递归的区别

【关键字：迭代，循环，遍历，递归的区别】【难度：初级】【掌握程度：了解】

解决方案：

1，循环（loop），指的是在满足条件的情况下，重复执行同一段代码。比如，while语句。循环则技能对应集合，列表，数组等，也能对执行代码进行操作。

2，迭代（iterate），指的是按照某种顺序逐个访问列表中的每一项。比如，for语句。迭代只能对应集合，列表，数组等。不能对执行代码进行迭代。

3，遍历（traversal），指的是按照一定的规则访问树形结构中的每个节点，而且每个节点都只访问一次。遍历同迭代一样，也不能对执行代码进行遍历。

4，递归（recursion），指的是一个函数不断调用自身的行为。比如，以编程方式输出著名的斐波纳契数列

## int转float为何会可能精度丢失的问题

【关键字：丢失精度】【难度：初级】【掌握程度：了解】

解决方案：Int是4字节32位来表示的，而float虽然也是4字节32位，但是float的存储结构是很不一样的，float的存储结构是1个符号位，8个指数位，23个尾数。float只能有24位来确定精度，而int是32位

## double类型为什么会出现精度误差

【关键字：double】【难度：初级】【掌握程度：了解】

解决方案：因为我们看到的十进制数在计算机底层都会转换成二进制，在转换的时候会出现误差

## char类型是两个字节，而中文汉字占3个字节，char是怎么保存中文汉字的？

【关键字：char】【难度：初级】【掌握程度：死背】

解决方案：char类型java中使用的是Unicode编码，中文汉字占两个字节，而UTF8编码中汉字占用3个字节。

## 创建对象时报错

【关键字：创建对象报错】【难度：初级】【掌握程度：了解】

解决方案：创建对象传的参数和类中定义的构造器不一样，所以出现编译错误

## 初始化数值问题

【关键字：初始化】【难度：初级】【掌握程度：了解】

解决方案：public class TestInstance {

public static TestInstance instance = new TestInstance();

public static int a;

public static int b = 0;

public TestInstance() {

a++;

b++;

}

public static void main(String[] args) {

System.out.println(TestInstance.a);

System.out.println(TestInstance.b);

}

}

学生不理解为什么运行结果是：a=1，b=0

之所以有这样的运行结果，这里涉及到类加载的顺序：

（1）在加载阶段，加载类的信息

（2）在准备阶段给instance、a、b做默认初始化并分配空间，此时a和b的值都为0

（3）在初始化阶段，执行构造方法，此时a和b的值都为1

（4）在初始化阶段，给静态变量做显式初始化，此时b的值为0

改一下代码的执行顺序，改成下面这个样子：

public class TestInstance {

public static int a;

public static int b = 0;

public static TestInstance instance = new TestInstance();

public TestInstance() {

a++;

b++;

}

public static void main(String[] args) {

System.out.println(TestInstance.a);

System.out.println(TestInstance.b);

}

}

运行结果是：a=1，b=0

这里涉及到类加载的顺序：

（1）在加载阶段，加载类的信息

（2）在准备阶段给instance、a、b做默认初始化并分配空间，此时a和b的值都为0

（3）在初始化阶段，给静态变量做显式初始化，此时b的值仍为0

（4）在初始化阶段，执行构造方法，此时a和b的值都为1

总结：类的初始化过程

TestInstance instance = new TestInstance();在内存中做了哪些事情?

（1）加载TestInstance.class文件进内存

（2）在栈内存为instance开辟空间

（3）在堆内存为TestInstance对象开辟空间

（4）对TestInstance对象的成员变量进行默认初始化

（5）对TestInstance对象的成员变量进行显示初始化

（6）通过构造方法对TestInstance对象的成员变量赋值

（7）TestInstance对象初始化完毕，把对象地址赋值给instance变量

## java到底是引用传递还是

【关键字：值传递和引用传递】【难度：初级】【掌握程度：了解】

解决方案：java使用对象的引用来做计算，所有的对象变量都是引用，但是在方法传递参数的时候传的是值

## 面向对象到底什么意思

【关键字：面向对象】【难度：初级】【掌握程度：了解】

解决方案：概念性的，讲师也讲到了。还不明白的学生，可以举例，用生活中直观的事物来说明。就比如你要坐车去某个地方，你要去哪，坐几路车，哪里下，哪里换乘，这是一个过程。但面向对象时，你可以把那名开车的司机和车作为对象，他们有个属性也就是功能，可以把你送到目的地。如果你是按面向过程的方式去目的地的话，那你就得按照那些步骤去，但如果是面向对象的话，你直接调用那名司机和车，他们就会帮你把送到目的地，而不用你关心怎么个去法，也就是去的过程。

其实，这跟当今社会分工一样。有司机，有医生，有老师，他们都有自己的功能，也就是JAVA上面的属性和方法，你可以作为一个调度者，直接调用派遣他们，完成你想做的事情。而不用去管自己该怎么去完成这个事情。比如建个房子，你可以调用工程队，但不用去管怎么个建造的过程。

这些概念需要长时间去领悟。

## This怎么用

【关键字：this】【难度：初级】【掌握程度：了解】

解决方案：This操作1种是指代当前对象，另外一种是this() 指代构造方法

## 构造方法和普通方法的区别

【关键字：构造方法】【难度：初级】【掌握程度：了解】

解决方案：构造方法是创建对象时调用的,普通方法是创建后的对象调用的

## this与实例化对象的区别？例如在构造方法中创建当前类的对象，通过对象调用属性并赋值，和this.属性赋值不同，为什么？

【关键字：this与实例化对象的区别】【难度：初级】【掌握程度：了解】

解决方案：this关键字一般应用于实例方法或构造方法中，代表一个引用，用于指向调用当前方法的对象。其用法：this代表实例方法/实例变量，this用在构造方法内部，用于访问本类的其他构造方法或者属性，所以代表的是当前对象，那每个对象构建之后都在堆内存中开辟块一块空间，每个对象的内存地址都是不相同的，所以对象指向的对内存也是不同的，琦内存中存储的属性值也是不同的，所以会出现值不同。

## Integer.toString(num)与String.valueof()的区别

【关键字：Integer.toString(num)与String.valueof()】【难度：初级】【掌握程度：了解】

解决方案：

1、String.valueOf(int i)其实是调用了Integer.toString(int i)方法。

2、JAVA 中int类型转String类型的通常方法,有三种:

1、String.valueOf(int i)

2、Integer.toString(int i)

3、i+""; //i 为 int类型

三种方法效率排序为：

Integer.toString(int i) > String.valueOf(int i) > i+"";

## int x=5,y=6,z=7; z=z+--y\*z++;System.out.println(z);为什么输出的是42

【关键字：计算方法】【难度：初级】【掌握程度：了解】

解决方案：.java文件编译成.class文件后，同一个表达式里，相同的变量，用的是相同的值。

## 关于猜字母游戏，如果输入5个相同的字母，得到的结果会出现问题

【关键字：猜字母游戏】【难度：初级】【掌握程度：了解】

解决方案：我们在generate()中生成的就是不重复的字母，所以逻辑就排除了存在重复字母的可能

## static方法的应用场景

【关键字：static】【难度：初级】【掌握程度：了解】

解决方案：静态方法在类加载的时候执行，且只加载一次，所以一些不会发生变化的内容就可以放到静态方法中执行。

## 静态方法为什么不能直接调用非静态方法

【关键字：静态方法】【难度：初级】【掌握程度：了解】

解决方案：从类加载和对象创建的顺序上考虑，类加载在创建对象之前，所以静态方法在非静态方法加载之前执行，故不能直接调用非静态方法。

## Eclipse 无法打断点问题---遇到带斜杠的断点图标

【关键字：eclipse】【难度：初级】【掌握程度：了解】

解决方案：在run菜单里面，把skip all breakpoints 选项勾去即可

## Override注解的含义

【关键字：Override】【难度：初级】【掌握程度：了解】

解决方案：该注解表示方法是重写方法，方法上有该注解代表该方法一定是重写方法，但是没有该注解，该方法也有可能是重写方法。

## super(),this()的用法

【关键字：super】【难度：初级】【掌握程度：了解】

解决方案：super()是调用父类无参构造方法，this()调用本类的无参构造方法，super()和this()都只能放在构造方法的第一行，而且两个不能同时存在同一个构造方法中。

如果构造方法中没有显式的super()和this() 那么会默认调用super()，不会调用this()。

## 引用类型和基本类型的区别

【关键字：引用类型和基本类型的区别】【难度：初级】【掌握程度：了解】

解决方案：除了8大基本类型的其他所有类型都是引用类型

## 重写和重载在使用时有什么区别

【关键字：重写和重载的区别】【难度：初级】【掌握程度：死背】

解决方案：重写重载在使用时，重在必须在一个类中，重写必须在有继承关系的子父类中

## 以下程序的运行效果

【关键字：父子类程序运行效果】【难度：初级】【掌握程度：了解】

class Father {

static { System.out.println("father init"); }

public static int a = 1;

}

class Son extends Father {

static { System.out.println("son init"); }

public static int b = 2;

}

public class JavaTest {

public static void main(String[] args) {

System.out.println(Son.a);

System.out.println(Son.b);

}

}

运行效果：

father init

1

son init

2

以上的son类做了如下的修改：

public class Son extends Father {

static { System.out.println("son init"); }

public static int b = 2;

public static final int c = 3;

}

public class JavaTest {

public static void main(String[] args) {

System.out.println(Son.c);

}

}

运行效果：

3

答：初始化阶段是执行类中定义的Java程序代码或者说是字节码

在以下几种情况中，会执行初始化过程：

（1）创建类的实例

（2）访问类或接口的静态变量（特例：如果是用static final修饰的常量，那就不会对类进行显式初始化。static final 修改的变量则会做显式初始化）

（3）调用类的静态方法

（4）反射（Class.forName(packagename.className)）

（5）初始化类的子类。注：子类初始化问题：满足主动调用，即父类访问子类中的静态变量、方法，子类才会初始化；否则仅父类初始化。

（6）java虚拟机启动时被标明为启动类的类

从以上规则可以看出由于c是final static修饰的静态常量，所以先初始化，所以根本就没有调用静态代码块里面的内容，也就是说，没有对这个类进行显式初始化。

## 在同一个包中分不同的.java文件进行继承实验时，出现编译错误：java错误找不到符号

【关键字：继承】【难度：初级】【掌握程度：了解】

代码：  
package test;  
class Test{  
  public static void main(String[] args){  
    Person p = new Person("lpp");  
    System.out.println(p);  
  }  
}  
package test;  
public class Person{  
  String name;  
  public Person(String name){  
    this.name = name;  
  }  
}

1，问题分析：  
出现这个问题是因为在编译的时候使用的格式为javac -d . Test.java

2，解决方案：  
编译时按照如下格式： javac -d . \*.java  
执行时按照如下格式：java test.Test

3，总结：  
对于同一个包下定义了多个.java文件的情况，在编译时需注意，使用javac -d . \*.java的方法可以对包下面所有public类型的class统一编译，若只对一个9特定类进行编译，例如javac -d . Test.java，则会出现“找不到符号”的错误，原因是其他public 类没有被编译成.class文件。

## **在设计接口，对于接口方法何时需要声明抛出受检异常或者说所有的接口方法最后都声明抛出受检异常？**

【关键字：接口异常处理】【难度：初级】【掌握程度：了解】

代码：  
public interface xx{  
　public void method();  
}  
public interface xx{  
　public void method() throws Exception;  
}  
（2），如果需要声明抛出受检异常，那是抛出一个抽象的异常呢;还是抛出多个具体的异常？  
抛出抽象的异常。如果是这样那是抛出自定义抽象异常呢，还是直接抛出Exception？  
自定义抽象异常的代码：  
public interface xx{  
　public void method() throws CustomAbstractException;  
}  
直接抛出Exception的代码：  
public interface xx{  
　public void method() throws Exception;  
}  
抛出具体异常的代码：  
public interface xx{  
　public void method() throws SpecificException1,SpecificException2,...;  
}

答：  
关于异常，这里需要注意重要的知识点：  
1、不能放大接口可能会抛出的异常：Java 中子类重写父类的方法时声明抛出异常不能比父类范围大 ，同理:实现类中的实现方法声明抛出的异常不能比接口或者抽象类中的范围大。  
2、不能缩小接口的可视性  
3、异常在向上抛得过程中，如果main方法也无法处理，jvm就会终止程序。  
接口就是标准，异常是返回结果之一。  
使用不同的异常区分不同错误的情况，在方法声明中抛出所以可能的异常，即抛出具体异常，只有在具体的实现类中比较可行  
在接口有多个不同的实现类时，不同的实现又可能抛出不同的异常，这样就无法在接口声明中将这些具体异常全部声明出来了，这时候使用一个抽象的异常就比较可行了。可是如果这样的话，我觉的使用接口的地方，就无法明确到底会出现哪些具体的异常了。这样貌似异常的使用原则有些相违背了。  
还有如果在接口中声明了抛出自定义的抽象异常，那么在实现类中将一些如例如网络可能中断，未知的程序错误等等，这种处理成运行时异常，不作声明比较好，接收的人也没办法做任何处理对于有业务性的错误：例如取款时，余额不够、权限不够、超出最大取款限制等等，这类错误，最好，每个场景，定义一个异常，在申明中逐一说明，总之，异常是帮助使用者了解、处理不同的错误场景。使用者可以根据错误类型的不同，给用户提供不同的处理方式和流程。当然，如果你提供给用户的只是提示信息，就没必要再区分类型了

## JDK1.7中，接口类只有常量和抽象方法，但是在接口中写带方法体的静态方法不会报错。

【关键字：JDK 特性】【难度：初级】【掌握程度：了解】

解决方案：Java8的新特性，接口中可以有默认方法和静态方法。

## 代码导入Eclipse乱码

【关键字：eclipse操作】【难度：初级】【掌握程度：了解】

解决方案：修改项目编码 点击项目-->右键属性--->把GBK改（utf-8）或UTF-8改成GBK格式

## **volatile关键字是否能保证线程安全？**

【关键字：volatile】【难度：中级】【掌握程度：了解】

解决方案：volatile关键字用在多线程同步中，可保证读取的可见性，JVM只是保证从主内存加载到线程工作内存的值是最新的读取值，而非cache中。但多个线程对volatile的写操作，无法保证线程安全。例如假如线程1，线程2 在进行read,load 操作中，发现主内存中count的值都是5，那么都会加载这个最新的值，在线程1堆count进行修改之后，会write到主内存中，主内存中的count变量就会变为6；线程2由于已经进行read,load操作，在进行运算之后，也会更新主内存count的变量值为6；导致两个线程及时用volatile关键字修改之后，还是会存在并发的情况。

## 静态变量、静态方法、静态块 继承问题？

【关键字：static关键字】【难度：初级】【掌握程度：掌握】

解决方案：1. 通过类名来调用子类中的静态变量和静态方法，当父类与子类相同时是，子类会隐藏父类中与其相同的静态变量和静态方法，如果子类中没有与其父类相同的静态变量和静态方法，子类从其父类调用过来的静态变量和静态方法就会表现出来。

2. 通过子类创建对象来用对象名调用子类中的静态变量和静态方法，除非是父类没有的静态变量和静态方法，会显示其子类的静态变量和静态方法。否则，最后显示一定是从父类哪里引用来的静态变量和静态方法。

3. 静态块不能被继承。

## 85. 如何关联jdk源码？

【关键字：JDK源码】【难度：初级】【掌握程度：了解】

解决方案：jdk安装包下有src.zip源码包，可以crtl鼠标左键查看源码，通过弹出的Attach Source选择src.zip压缩包，即可关联源码

## instanceof关键字是什么意思，怎么理解？

【关键字：instanceof关键字】【难度：中级】【掌握程度：掌握】

解决方案：该关键字是在强制类型转换时进行使用，一般是用来合理规划逻辑严谨性，所以在强制类型转换之前最好加上类型判断，同时该关键字一般是针对引用类型变量的。

## **String字符问题**

String s1="ABC";

String s2="A"+"BC";

String s3=new String ("ABC");

在此过程中创建了几个对象？

【关键字：String字符串】【难度：中级】【掌握程度：掌握】

解决方案：创建了两个：String s1="ABC"; 在常量池中创建了一个对象，String s2="A"+"BC";

在编译时被编译为String s1="ABC";会重用常量池中的“ABC”对象，String s3=new String ("ABC");会重用常量池中的“ABC”；另外在堆中创建“ABC”对象

## 对象无法点出来相应的方法？

【关键字：引用】【难度：初级】【掌握程度：了解】

解决方案：在定义类的时候创建对象的名字一样，那就相当于类进行了重新定义，所以无法点出相应的方法。

## 子类里面写无参构造方法报错，父类不是默认有无参构造方法吗？

【关键字：够造器】【难度：中级】【掌握程度：了解】

解决方案：当没有构造方法是，父类是会有默认无参构造方法，如果父类写了有参构造方法那么就会被重写，父类的无参构造方法就会被去掉；只要在父类手动添加无参构造方法就好了

## 自增丢失问题详解

【关键字：自增键值】【难度：初级】【掌握程度：了解】

解决方案：a=++a和 a=a++的区别  
在C中a=a++是等1 的  而在 java中a是等于0  
int a=0;a=a++;  
生成的字节码是：  
0 iconst\_0 -----0压入栈顶  
1 istore\_1 [a] -----栈顶值（0）存入局部变量1（a）  
2 iload\_1 [a] -----变量1的值压入栈顶（值为0）(a++效果)  
3 iinc 1 1 [a] -----变量1自加（a++操作，自加后变量1的值为1）  
6 istore\_1 [a] -----将栈顶值（0）存入变量1（=操作）  
a的值依次变化为：0->1->0  
跟学员解释就是   a++ 代表再下一次见到a时给a加1  此时 a值为0  赋值给a  所以a=0

## [\\+86](file://+86) 正则表达式 [\\是干嘛用的？](file:///\\\\是干嘛用的？)

【关键字：正则表达式】【难度：初级】【掌握程度：掌握】

解决方案：在字符串中+是字符连接符,对+进行转译需要加\,而对\转译也需要加\,所以最终结果是\\+

## break和exit()的区别？

【关键字：break和exit关键字】【难度：初级】【掌握程度：了解】

解决方案：break语句的使用场合主要是switch语句和循环结构。在循环结构中使用break语句，如果执行了break语句，那么就退出循环，接着执行循环结构下面的第一条语句。调用exit()函数将会结束当前进程，当exit()中的参数为0时，表示正常退出，其他返回值表示非正常退出。

## final 修饰的StringBuilder对象，当调用append()方法，直到StringBuilder扩容之后。内部调用Arrays.copyof()方法创建新的字符数组，为什么final还不报错？

【关键字：final关键字】【难度：初级】【掌握程度：了解】

解决方案：因为那个字符数组只是StringBuilder对象里面的一个属性，属性变了对象还没变，不报错。

## 类方法和实例方法是什么？

【关键字：方法应用】【难度：初级】【掌握程度：了解】

解决方案：类方法是静态方法，实例方法是对象引用的方法。

## 95. 将对象加入到集合中之后，将原有的引用设置为null之后，为何输出集合，还是能输出之前的对象？

【关键字：集合操作】【难度：中级】【掌握程度：掌握】

解决方案：集合中存储的是对象的引用，将原有引用加入到集合中，相当于是把该对象的引用复制了一份到集合中，虽然原有引用为null了，但是集合中的复制的引用仍然有用。

## 96.常量池放在什么位置？

【关键字：内存分布】【难度：初级】【掌握程度：了解】

解决方案：内存分为：堆，栈，方法区，那常量池是在堆里面存储。

## 97. 为什么今天讲的person的输出一定要使用toString方法输出，不直接输出就好，结果都一样？

【关键字：重写toString】【难度：初级】【掌握程度：了解】

解决方案：toString方法，如果在类中有写，那么就会调用类中的这个方法，所有对象都继承Object，如果类中没有toString方法，那么在输出时会自动调用Object的toString方法输出为xxxx@xxxxxxx的类加地址形式。不管System.out.println(xxx.toString)这里如果xxx不是字符串类型不管有没有toString都会调用这个方法进行输出。

## 98. 使用集合自带的remove方法删除元素，出错。

【关键字：集合remove】【难度：初级】【掌握程度：了解】

解决方案：集合的长度会根据元素个数而改变，使用集合自带的remove删除，会出现异常，需要使用迭代器对集合进行删除操作

## 99. 关于正则表达式中的^和$的使用有点模糊

【关键字：正则表达式】【难度：初级】【掌握程度:了解】

解决方案： ^是用在正则开头，$用在正在末尾，加了这两个表示，开头与结尾必须按照正则表达式中的匹配

## 100. Collection<String> c = new ArrayList<>();为什么右边没有写泛型String？

【关键字：集合泛型】【难度：初级】【掌握程度: 掌握】

解决方案：在JDK1.8之后，后面可以默认不写，类型和左边第一次定义的相同。都是String

## 101. Java引用有几种状态？

【关键字：引用】【难度：初级】【掌握程度: 了解】

解决方案：

强引用：用的最广。我们平时写代码时，new一个Object存放在堆内存，然后用一个引用指向它，这就是强引用。如果一个对象具有强引用，那垃圾回收器绝不会回收它。当内存空间不足，Java虚拟机宁愿抛出OutOfMemoryError错误，使程序异常终止，也不会靠随意回收具有强引用的对象来解决内存不足的问题。  
  
 软引用：如果一个对象只具有软引用，则内存空间足够时，垃圾回收器就不会回收它；如果内存空间不足了，就会回收这些对象的内存。只要垃圾回收器没有回收它，该对象就可以被程序使用。软引用可用来实现内存敏感的高速缓存。  
  
 弱引用：弱引用与软引用的区别在于：只具有弱引用的对象拥有更短暂的生命周期。每次执行GC的时候，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。不过，由于垃圾回收器是一个优先级很低的线程，因此不一定会很快发现那些只具有弱引用的对象。  
  
 虚引用：“虚引用”顾名思义，就是形同虚设，与其他几种引用都不同，虚引用并不会决定对象的生命周期。如果一个对象仅持有虚引用，那么它就和没有任何引用一样，在任何时候都可能被垃圾回收器回收。虚引用主要用来跟踪对象被垃圾回收器回收的活动。

## 102. Java中的内存怎么划分的？

【关键字：引用】【难度：初级】【掌握程度: 了解】

解决方案：jvm运行时的状态。具体划分为如下5个内存空间：  
程序计数器：保证线程切换后能恢复到原来的执行位置  
虚拟机栈：为虚拟机执行java方法服务：方法被调用时创建栈帧-->局部变量表->局部变量、对象引用  
本地方法栈：为虚拟机执使用到的Native方法服务  
堆内存：存放所有new出来的东西  
方法区：存储被虚拟机加载的类信息、常量、静态常量、静态方法等。  
运行时常量池，内存区域中的程序计数器、虚拟机栈、本地方法栈这3个区域随着线程而生，线程而灭；栈中的栈帧随着方法的进入和退出而有条不紊地执行着出栈和入栈的操作，每个栈帧中分配多少内存基本是在类结构确定下来时就已知的。在这几个区域不需要过多考虑回收的问题，因为方法结束或者线程结束时，内存自然就跟着回收了

## 103. break与 continue的区别？

【关键字：break】【难度：初级】【掌握程度: 了解】

解决方案：

1、二者的相同点：都可以用在循环语句中，break还可用在switch中。

2、不同点：break可以跳出当前的循环中，即结束本次循环（本次循环已经终止）

  continue：只能终止本次循环，还需进行下一轮的循环（本次循环还会继续）

## 104. 值传递和引用传递有什么区别？

【关键字：传递】【难度：初级】【掌握程度：了解】

解决方案：值传递传递的是基本数据类型的值，而引用传递传递的是对象的地址值。

## 105.java中子类是否可以继承父类的静态变量和方法并呈现出多态特性。

【关键字：继承】【难度：初级】【掌握程度：掌握】

解决方案：1）子类是不继承父类的static变量和方法的。因为这是属于类本身的。但是子类是可以访问的。2）子类和父类中同名的static变量和方法都是相互独立的，并不存在任何的重写的关系。

## 106. 为甚什么父类写了有参构造函数，子类就会提示要求定义父类的无参构造或者是子类的有参构造函数中有父类有参构造函数的变量。

【关键字：构造器】【难度：初级】【掌握程度：掌握】

解决方案：

 1.如果父类有默认构造器(构造器没有参数),则子类随意  
   2.如果父类没有默认构造器,但是有其他的构造器(有参数),则子类必须要写构造器   
 因为父类已经定义了一个有参的构造函数，此时编译器不会为你调用默认的构造数，  
 当子类继承时，必须在自己的构造函数显式调用父类的构造函数，自己才能确保子类在初始化前父类会被实例化，如果你父类中有无参的构造函数，子类就不会强制要求调用，即你写的那个就可以通过，编译器会默认帮你调用父类的构造函数

## 107. Socket是什么概念

【关键字：Socket】【难度：初级】【掌握程度：掌握】

解决方案：

Socket是两台主机之间的一个连接，它可以完成7个基本操作：  
 1）连接远程主机  
 2）发送数据  
 3）接收数据  
 4）关闭连接  
 5）绑定端口  
 6）监听入站数据  
 7）在绑定端口上接受来自远程机器的连接  
一旦建立了socket连接，就可以使用输入输出流，这个连接是全双工的（full-duplex），两台主机都可以同时发送和接收数据。

## 108.关于使用File操作文件的时候，普通文件可以读取，传入盘符就会报空指针异常？

【关键字：File文件操作】【难度：初级】【掌握程度：掌握】

解决方案：由于盘符内含有部分隐藏文件，并且权限是不可读的，所以会报错，需要做个判断，不读取这些文件就可以了

## 109. if判断语句后面的大括号有与没有有什么区别

【关键字：if选择条件】【难度：初级】【掌握程度：掌握】

解决方案：有大括号的时候 大括号里面所有的 都归if管。只有条件为真的时候  才会执行。没有大括号的时候 只有下面的一句归if管

## 110. 8>>>2=?>>>是个什么符号？

【关键字：位运算符】【难度：初级】【掌握程度：了解】

解决方案：java二进制位运算符，右移2位，也就是2的2次方

## 111. java为什么匿名内部类的参数引用是final修饰符？

【关键字：匿名内部类】【难度：初级】【掌握程度：了解】

解决方案：

以下代码中为什么匿名内部类的参数引用是final？  
 interface AnnoInner(){addXYZ();}  
 public class Outer {  
  public AnnoInner getAnnoInner(final int x){  
    final int y=100;  
    return new AnnoInner(){  
    int z=100;  
    public int addXYZ(){return x+y+z;}  
 //public void changeY(){y+=1;} 这个函数无法修改外部环境中的自由变量y。  
    };  }  
  private int num=100;  
}  
匿名内部类因为是匿名，所以不能显式地声明构造函数，也不能往构造函数里传参数。不但返回的只是个叫AnnoInner的接口，而且还没有和它外围环境getAnnoInner()方法的局部变量x和y构成任何类的结构。但它的addXYZ()函数却直接使用了x和y这两个自由变量来计算结果。说明外部方法getAnnoInner()事实上已经对内部类AnnoInner构成了一个闭包。  
但这里奇怪的地方是这两个x和y都必须用final修饰，不可以修改。如果用一个changeY()函数试图修改外部getAnnoInner()函数的变量y，编译器通不过  
会报出：error: cannot assign a value to final variable y 错误。因为这里Java编译器支持了闭包，但支持地不完整。因为编译器编译的时候其实悄悄对函数做了手脚，偷偷把外部方法的x和y局部变量，拷贝了一份到匿名内部类里。如以下代码  
interface AnnoInner(){addXYZ();}  
public class Outer {  
  public AnnoInner getAnnoInner(final int x){  
    final int y=100;  
    return new AnnoInner(){  
      int copyX=x; //编译器相当于拷贝了外部自由变量x的一个副本到匿名内部类里。  
      int copyY=y; //编译器相当于拷贝了外部自由变量y的一个副本到匿名内部类里。  
      int z=100;  
      public int addXYZ(){return copyX+copyY+z;}  
};}  
所以Java编译器实现的只是capture-by-value，并没有实现capture-by-reference。而只有后者才能保持匿名内部类和外部环境局部变量保持同步。但Java实现的只是capture-by-value所以就不许大家改外围的局部变量

## 112. 正则表达式有+和没有+的区别？

【关键字：正则表达式】【难度：初级】【掌握程度：了解】

解决方案：后面有+代表前面的表达式会可能匹配到一个或者多个

## 115. RandomAccessFile在递归的情况下，方法中无法关闭，释放内存。

【关键字：Linux命令】【难度：初级】【掌握程度：了解】

解决方案：可以将它设置成全局变量，当作参数传入方法中，然后在方法外关闭。

## 116. 什么是运行时确定，什么是编译时确定？

【关键字：java编译器】【难度：初级】【掌握程度：了解】

解决方案：编译时就是在代码编译成.class文件时就能确定的，比如int类型的加减法，String类型的字符串拼接，在编译时就计算出了结果，不占用运行时的效率。运行时就是编译时不能确定的，当你在表达式中引用了变量

## 117. 查看目录中的文件，如果过多，怎么使用分页显示？

【关键字：Linux命令】【难度：初级】【掌握程度：了解】

解决方案：

有些命令显示的内容太多，一页屏幕显示不完，就需要翻页命令。使用方法一般是用管道符导出这样的命令的输出，来让more或less处理，并一页一页的查看，就像这样：  
 ls -al /etc |more （/etc目录的文件太多）  
 或ls -al /etc |less  
 命令说明，more命令的翻页按键：  
 空格：表示向下翻一页  
 回车：表示向下滚动一行  
 /字符串：表示查询字符串

q：离开more界面

## 118. FiilInputStream在创建实例时,有两个构造方法,一个是传入一个字符串对象,一个是传入一个File对象,有什么区别

【关键字：文件流】【难度：初级】【掌握程度：了解】

解决方案：FiilInputStream 在传入字符串时,参数为字符串的构造方法会在内部创建File对象进行操作,传入File对象时,直接进行操作

## 119. HashMap的工作原理

【关键字：Map集合】【难度：初级】【掌握程度：掌握】

解决方案：HashMap在Map.Entry静态内部类实现中存储key-value对。HashMap使用哈希算法，在put和get方法中，它使用hashCode()和equals()方法。当我们通过传递key-value对调用put方法的时候，HashMap使用Key hashCode()和哈希算法来找出存储key-value对的索引。Entry存储在LinkedList中，所以如果存在entry，它使用equals()方法来检查传递的key是否已经存在，如果存在，它会覆盖value，如果不存在，它会创建一个新的entry然后保存。当我们通过传递key调用get方法时，它再次使用hashCode()来找到数组中的索引，然后使用equals()方法找出正确的Entry，然后返回它的值。

## 120. break,return,continue的区别不是太清楚，都怎么用？

【关键字：关键字的使用】【难度：初级】【掌握程度：掌握】

解决方案：break用于跳出循环体，结束循环；return用于结束方法，在循环语句中也可以使用，直接结束整个方法；continue结束本次循环，而进入下次循环

## 121. 为什么FileInputStream有的时候读取文件时不能完全的读取完文件中的数据。报NullpointException？

【关键字：IO流】【难度：初级】【掌握程度：掌握】

解决方案：byte[]里面的实际参数不是一个byte的数据大小，而是文件的长度 改过后就不会报错 。

## 122. swing中怎么准确获取到鼠标的坐标值

【关键字：Swing】【难度：初级】【掌握程度：了解】

解决方案：mouseClick(MouseEvent e) 可以直接获取：e.getX();e.getY();

## 123. 为什么double i=1; i/=3;i\*=3;输出i还是1，而不是0.99999？即精度为何没有丢失？如果把3换成103，就会丢失精度，输出0.999999

【关键字：基本类型精确度】【难度：初级】【掌握程度：了解】

解答方案： 未解决

## 123.exsits中关联的条件有什么讲究？

【关键字：exsits】【难度：初级】【掌握程度：了解】

解决方案：关系到exists的原理，其实就是用外表的记录不断loop判断取值。

## 124. 对double的精度丢失不大了解

【关键字：基本类型精确度】【难度：初级】【掌握程度：了解】

解决方案：给他讲解运算的实现流程

## 125. nullpointException异常？

【关键字：空指针异常】【难度：初级】【掌握程度：了解】

解决方案：这样的原因是因为没有在驱动的时候初始化 ，或者和自表中的字段不对应造成的。

## 126. 聊天室案例ChatProtocol类里面DataInputStream和DataOutputStream是私有化的属性，子类客户端和服务端继承不了，那他是怎么传数据的。

【关键字：继承】【难度：初级】【掌握程度：了解】

解决方案：子类并没有直接操作这两个属性，创建子类对象的时候会创建父类。子类调用的是父类公共的方法，具体初始化DataInputStream和DataOutputStream是再父类中

## 127. 怎么检测一个线程是否持有对象监视器

【关键字：线程】【难度：初级】【掌握程度：了解】

解决方案：学生问一道多线程面试题：怎么检测一个线程是否持有对象监视器。有方法可以判断某个线程是否持有对象监视器，Thread类提供了一个holdsLock(Object obj)方法，当且仅当对象obj的监视器被某条线程持有的时候才会返回true，注意这是一个static方法，这意味着"某条线程"指的是当前线程。

## 128. synchronized和ReentrantLock的区别

【关键字：线程】【难度：初级】【掌握程度：了解】

解决方案：synchronized是和if、else、for、while一样的关键字，ReentrantLock是类，这是二者的本质区别。既然ReentrantLock是类，那么它就提供了比synchronized更多更灵活的特性，可以被继承、可以有方法、可以有各种各样的类变量，ReentrantLock比synchronized的扩展性体现在几点上：  
1、ReentrantLock可以对获取锁的等待时间进行设置，这样就避免了死锁  
2、ReentrantLock可以获取各种锁的信息  
3、ReentrantLock可以灵活地实现多路通知  
另外，二者的锁机制其实也是不一样的。ReentrantLock底层调用的是Unsafe的park方法加锁，synchronized操作的应该是对象头中mark word

## 129.在关闭resultset的时候，为什么要不等于null才关闭？

【关键字：jdbc】【难度：初级】【掌握程度：了解】

解决方案：关闭resultset是调用的close方法，如果resultset为空的话，调用close方法会报空指针异常。

## 130. mysql数据库指定文件找不到？

【关键字：数据库】【难度：初级】【掌握程度：了解】

解决方案：应该是卸载不干净引起的，已经知道重装。

## 131. 关于使用maven导入jar包的原理，以及普通项目如何导如jar包

【关键字：maven】【难度：初级】【掌握程度：了解】

解决方案：maven相当于是一个仓库，里面有很多jar包，只要确定索引，就会从maven仓库下载jar包，普通项目如果想导入jar包的话，需要先在网上下载好jar包，然后通过bulid path-->add jars导入jar包

## 132. 很多学员按照mysql不仔细，编码集合没有设置为utf8，导致数据插入不进去或者插入进去是乱码情况

【关键字：mysql】【难度：初级】【掌握程度：了解】

解决方案：更改mysql配置文件或者重装mysql

## 133. 反射在实际工作中的运用

【关键字：反射】【难度：初级】【掌握程度：了解】

解决方案：Java动态代理和学习框架时,原理都是使用反射技术实现所需功能

## 134. 毫秒值,Date,Calendar之间的相互转化

【关键字：日期类型转换】【难度：初级】【掌握程度：了解】

解决方案：通过Date的构造方法,SimpleDateFormat工具类等API,进行三者之间的相互转化,实现对日期的具体操作

## 135. 触发器创建失败

【关键字：数据库】【难度：初级】【掌握程度：了解】

解决方案：回头查了一下，是因为库里面已经创建过了，删除之后就可以了。

## 136. maven环境问题

【关键字：maven】【难度：初级】【掌握程度：了解】

解决方案：配置问题，settings文件没有覆盖

## 136. sleep方法和wait方法有什么区别

【关键字：线程】【难度：初级】【掌握程度：了解】

解决方案：sleep方法和wait方法都可以用来放弃CPU一定的时间，不同点在于如果线程持有某个对象的监视器，sleep方法不会放弃这个对象的监视器，wait方法会放弃这个对象的监视器

## 137. 生产者消费者模型的作用是什么

【关键字：】【难度：初级】【掌握程度：了解】

解决方案：1、通过平衡生产者的生产能力和消费者的消费能力来提升整个系统的运行效率，这是生产者消费者模型最重要的作用  
2、解耦，这是生产者消费者模型附带的作用，解耦意味着生产者和消费者之间的联系少，联系越少越可以独自发展而不需要收到相互的制约

## 138. ThreadLocal有什么用

【关键字：线程】【难度：初级】【掌握程度：了解】

解决方案：ThreadLocal就是一种以空间换时间的做法，在每个Thread里面维护了一个以开地址法实现的ThreadLocal.ThreadLocalMap，把数据进行隔离，数据不共享，自然就没有线程安全方面的问题了

## 139. Lamdba表达式 简洁写法

【关键字：】【难度：初级】【掌握程度：了解】

解决方案：Test t= (a) -> {return a + 1;}; 标准写法

Test t = a -> a + 1;  简洁写法，一个参数可以不写(),方法体一行代码可以不写 return 和 {}

## 140. windows环境 mysqldump -u root -p db1 > E:/aa.sql;报错

【关键字：mysql数据库】【难度：初级】【掌握程度：了解】

解决方案：学员在mysql环境中插入，应该在cmd环境

## 141. 疑问：海涛老师笔记，mysql-day07有 \* @表示回话变量  \* @@表示全局变量，指定是系统变量 但是，本人测试set @@autocommit=OFF； 该语句是设置当前会话连接的自动提交关闭，只针对当前会话连接，连接结束之后就会失效。 SET @@global.autocommit=OFF; 该语句是设置MySql的参数配置，全局配置，相当于关闭所有数据库的提交模式，设置成功之后必须重启客户端才能生效。 而且永久保存 问：@@在该测试过程，仿佛是session范围？？？

【关键字：mysql数据库】【难度：初级】【掌握程度：了解】

解决方案：已经解决  
set @@autocommit=OFF；是将当前会话的系统变量设置为off  
SET @@global.autocommit=OFF;该语句是设置MySql的参数配置，全局配置，

## 142. 为什么两台电脑，创建了用户，授权也完成但是不能访问对方的数据库？

【关键字：数据库】【难度：初级】【掌握程度：了解】

解决方案：ping一下，测试网络是否连接。事实上，没有，打开防火墙就好了。

## 143. 如何设置保存点？如何回滚到保存点。

【关键字：数据库】【难度：初级】【掌握程度：了解】

解决方案：Savepoint a。Roll back to a。即可。

## 144. 相同的字符串为什么可能会有不同的HashCode？

【关键字：字符】【难度：初级】【掌握程度：了解】

解决方案：如果自己重写equals和hashcode方法是可以实现的，但是java规范要求两个equals的对象一定要有相同的hashcode

## 145. 列举一个死锁的例子

【关键字：线程】【难度：初级】【掌握程度：了解】

解决方案：一个简单的死锁类

当DeadLock类的对象flag==1时（td1），先锁定o1,睡眠500毫秒

而td1在睡眠的时候另一个flag==0的对象（td2）线程启动，先锁定o2,睡眠500毫秒

td1睡眠结束后需要锁定o2才能继续执行，而此时o2已被td2锁定；

td2睡眠结束后需要锁定o1才能继续执行，而此时o1已被td1锁定；

td1、td2相互等待，都需要得到对方锁定的资源才能继续执行，从而死锁。

## 146. 对char\_length(  )和length（）方法的不理解

【关键字：数据库】【难度：初级】【掌握程度：了解】

解决方案：一、char\_length(str)  
1、返回值为字符串str 的长度，长度的单位为字符，一个多字节字符算作一个单字符  
2、不管汉字还是数字或者是字母都算是一个字符  
3、任何编码下，多字节字符都算是一个字符  
二、length(str)  
1、utf8编码下,一个汉字是算三个字符,一个数字或字母算一个字符。  
2、其他编码下,一个汉字算两个字符, 一个数字或字母算一个字符。

## 147. scanner的nextLine和nextInt区别

【关键字：java基础】【难度：初级】【掌握程度：了解】

解决方案：next不会读取换行符，如果换行，下面的一个nextLine会读取到换行符导致数据为空

## 148. 练习题2中不使用round方法做数据精确取值查询不到数据

【关键字：】【难度：初级】【掌握程度：了解】

解决方案：原因是因为平均值avg(salary)中有除不尽的数值，记过不精确在做having过滤的时候找不到匹配的数据，导致查询结果为空，所以需要使用round对小数精确取值

## 149. volatile关键字的作用

【关键字：】【难度：初级】【掌握程度：了解】

解决方案：1、多线程主要围绕可见性和原子性两个特性而展开，使用volatile关键字修饰的变量，保证了其在多线程之间的可见性，即每次读取到volatile变量，一定是最新的数据  
2、代码底层执行不像我们看到的高级语言----Java程序这么简单，它的执行是Java代码-->字节码-->根据字节码执行对应的C/C++代码-->C/C++代码被编译成汇编语言-->和硬件电路交互，现实中，为了获取更好的性能JVM可能会对指令进行重排序，多线程下可能会出现一些意想不到的问题。使用volatile则会禁止语义重排序，当然这也一定程度上降低了代码执行效率

## 150. 死循环、死锁、阻塞、页面打开慢等问题

【关键字：线程】【难度：初级】【掌握程度：了解】

解决方案：死循环、死锁、阻塞、页面打开慢等问题，打线程dump是最好的解决问题的途径。所谓线程dump也就是线程堆栈，获取到线程堆栈有两步：  
1、获取到线程的pid，可以通过使用ps命令，在Linux环境下还可以使用ps -ef | grep java  
2、打印线程堆栈，可以通过使用stack pid命令，在Linux环境下还可以使用kill -3 pid  
另外提一点，Thread类提供了一个getStackTrace()方法也可以用于获取线程堆栈。这是一个实例方法，因此此方法是和具体线程实例绑定的，每次获取获取到的是具体某个线程当前运行的堆栈

## 151. 如何在两个线程之间共享数据

【关键字：线程】【难度：初级】【掌握程度：了解】

解决方案：通过在线程之间共享对象就可以了，然后通过wait/notify/notifyAll、await/signal/signalAll进行唤起和等待，比方说阻塞队列BlockingQueue就是为线程之间共享数据而设计的

## 152. varchar和char的区别

【关键字：数据库类型】【难度：初级】【掌握程度：掌握】

解决方案：主要他们暂用内存的区别，varchar按照实际内容暂用内存，char按照定义字节长度暂用内存。

## 153. 关于基本数据类型的转换和溢出问题

【关键字：基本类型转换】【难度：初级】【掌握程度：掌握】

解决方案：需要注意强转及数据类型的取值范围

## 154. http和https的区别？

【关键字：传输协议】【难度：初级】【掌握程度：掌握】

解决方案：

HTTP协议传输的数据都是未加密的，也就是明文的，因此使用HTTP协议传输隐私信息非常不安全，为了保证这些隐私数据能加密传输，于是网景公司设计了SSL（Secure Sockets Layer）协议用于对HTTP协议传输的数据进行加密，从而就诞生了HTTPS。简单来说，HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全。

## 155. Select StuId,StuName From tblStudent st Where not exists(Select CourseID From tblCourse cu Inner Join tblTeacher tc On cu.TeaID=tc.TeaID

## Where tc.TeaName='叶平' And CourseID not in (Select CourseID From tblScore Where StuID=st.StuID))这个答案如何理解的问题。

【关键字：数据库】【难度：初级】【掌握程度：掌握】

解决方案：not exists中第一个条件是：查询素有叶老师教过的课程，简称A，第二个条件是：courseID not in（所有学生选修过的课程ID）。简称B；已知叶老师肯定是教过至少一门课程的，因此A==true；至于B，not in(B==true),当！B==true的时候，说明有人没有上叶平老师的课程，此情况子句的整体逻辑为：not exists (true && true),!true==false，以此类推没有选修叶老师课程的学生，不会被查询出来。过滤条件直接否定了，所有没选修叶平老师课程的学生，包含一门课程都没有选修的学生。据目前看来，虽然很绕，但是sql执行效率应该是最高的，其他用复杂关联查询未必可以写出来，写出来效率也不高。

## 156. 学生在创建mysql函数时，参数个数不对

【关键字：数据库】【难度：初级】【掌握程度：掌握】

解决方案：

创建mysql函数时，函数名用了系统名的函数名，调用无效

## 157. 使用对象输出流，将对象写入硬盘里，为什么打开那个文件是一堆乱码？

【关键字：数据库】【难度：初级】【掌握程度：掌握】

解决方案：对象序列化之后，乱码是正常的。那一堆乱码就是这个对象。

## 158. nextLine()和next()的区别

【关键字：next关键字】【难度：初级】【掌握程度：掌握】

解决方案：**首先，next（）一定要读取到有效字符后才可以结束输入，对输入有效字符之前遇到的空格键、Tab键或Enter键等结束符，next（）方法会自动将其去掉，只有在输入有效字符之后，next（）方法才将其后输入的空格键、Tab键或Enter键等视为分隔符或结束符。简单地说，next（）查找并返回来自此扫描器的下一个完整标记。完整标记的前后是与分隔模式匹配的输入信息，所以next方法不能得到带空格的字符串而nextLine（）方法的结束符只是Enter键，即nextLine（）方法返回的是Enter键之前的所有字符，它是可以得到带空格的字符串的。**

## 159. 装饰模式与继承的区别

【关键字：设计模式】【难度：初级】【掌握程度：掌握】

解决方案：所谓模式，最大的功能就是更好的应对变化，当需要修改代码时，尽量少的去修改原有代码或者只通过增加类或者配置就可以完成变更，最大程度的保全原有结构，这个才是模式的精神。  
装饰者类拥有被装饰者类的对象，一般是当构造参数传入。  
在装饰者类当中调用被装饰者类的方法，封装成新的功能方法。 [装饰模式](https://www.baidu.com/s?wd=%E8%A3%85%E9%A5%B0%E6%A8%A1%E5%BC%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ujmvuhcsmW6smH0Lmvfd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1cvn163nHfvP1bsnjDvP1cd" \t "_blank)以对客户透明的方式动态地给一个对象附加上更多的责任。换言之，客户端并不会觉得对象在装饰前和装饰后有什么不同。[装饰模式](https://www.baidu.com/s?wd=%E8%A3%85%E9%A5%B0%E6%A8%A1%E5%BC%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ujmvuhcsmW6smH0Lmvfd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1cvn163nHfvP1bsnjDvP1cd" \t "_blank)可以在不使用创造更多子类的情况下，将对象的功能加以扩展。  


## 160. 数据库使用addBatch()方法，传入的参数可以是多条sql吗？

【关键字：addBatch】【难度：初级】【掌握程度：掌握】

解决方案：一次addBatch()只能传一条sql，不能一起传。

## 161. String str = new String("abc")创建了几个对象

【关键字：String】【难度：初级】【掌握程度：掌握】

解决方案：因为有new，所以堆中必然有一个对象。另外，如果常量池中已有"Hello World"，则不创建，没有，则在常量池中创建

## 162. 高并发、任务执行时间短的业务怎样使用线程池？并发不高、任务执行时间长的业务怎样使用线程池？并发高、业务执行时间长的业务怎样使用线程池？

【关键字：并发】【难度：初级】【掌握程度：掌握】

解决方案：（1）高并发、任务执行时间短的业务，线程池线程数可以设置为CPU核数+1，减少线程上下文的切换  
（2）并发不高、任务执行时间长的业务要区分开看：  
　a）假如是业务时间长集中在IO操作上，也就是IO密集型的任务，因为IO操作并不占用CPU，所以不要让所有的CPU闲下来，可以加大线程池中的线程数目，让CPU处理更多的业务  
　　b）假如是业务时间长集中在计算操作上，也就是计算密集型任务，这个就没办法了，和（1）一样吧，线程池中的线程数设置得少一些，减少线程上下文的切换  
（3）并发高、业务执行时间长，解决这种类型任务的关键不在于线程池而在于整体架构的设计，看看这些业务里面某些数据是否能做缓存是第一步，增加服务器是第二步，至于线程池的设置，设置参考（2）。最后，业务执行时间长的问题，也可能需要分析一下，看看能不能使用中间件对任务进行拆分和解耦。

## 163. win10系统的IE11不能使用HttpWatch 插件

【关键字：软件安装】【难度：初级】【掌握程度：掌握】

解决方案：载HttpWatch Version 11.0.23版本 支持

IE11  https://www.httpwatch.com/download/

## 164. 怎么检测一个线程是否持有对象监视器

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：有方法可以判断某个线程是否持有对象监视器，Thread类提供了一个holdsLock(Object obj)方法，当且仅当对象obj的监视器被某条线程持有的时候才会返回true，注意这是一个static方法，这意味着"某条线程"指的是当前线程。

## 165. synchronized和ReentrantLock的区别

【关键字：锁机制】【难度：初级】【掌握程度：掌握】

解决方案：synchronized是和if、else、for、while一样的关键字，ReentrantLock是类，这是二者的本质区别。既然ReentrantLock是类，那么它就提供了比synchronized更多更灵活的特性，可以被继承、可以有方法、可以有各种各样的类变量，ReentrantLock比synchronized的扩展性体现在几点上：  
1、ReentrantLock可以对获取锁的等待时间进行设置，这样就避免了死锁  
2、ReentrantLock可以获取各种锁的信息  
3、ReentrantLock可以灵活地实现多路通知  
另外，二者的锁机制其实也是不一样的。ReentrantLock底层调用的是Unsafe的park方法加锁，synchronized操作的应该是对象头中mark word

## 166. 数据库视图与表的关系，删了视图字段为什么表也跟着删除了字段

【关键字：视图】【难度：初级】【掌握程度：掌握】

解决方案：视图映射了表

## 167. public static void main(String args[]) {     int x=1,y=1,z=1;     if  (x--==1&&y++==1||z++==1)        System.out.println("x="+x+",y="+y+",z="+z);   } 问输出结果？

【关键字：逻辑运算符】【难度：初级】【掌握程度：掌握】

解决方案：考察短路|   z++==1 并未执行   所以结果为0，2，1

## 168. 静态方法的父子类继承关系

【关键字：静态方法】【难度：初级】【掌握程度：掌握】

解决方案：

父子类中两个一模一样的方法不属于继承方法，没有关系

## 169. Java里自旋是什么？

【关键字：synchronized】【难度：初级】【掌握程度：掌握】

解决方案：

很多synchronized里面的代码只是一些很简单的代码，执行时间非常快，此时等待的线程都加锁可能是一种不太值得的操作，因为线程阻塞涉及到用户态和内核态切换的问题。既然synchronized里面的代码执行得非常快，不妨让等待锁的线程不要被阻塞，而是在synchronized的边界做忙循环，这就是自旋。如果做了多次忙循环发现还没有获得锁，再阻塞，这样可能是一种更好的策略

## 170. 什么是乐观锁和悲观锁

【关键字：线程锁机制】【难度：中级】【掌握程度：掌握】

解决方案：

（1）乐观锁：就像它的名字一样，对于并发间操作产生的线程安全问题持乐观状态，乐观锁认为竞争不总是会发生，因此它不需要持有锁，将比较-替换这两个动作作为一个原子操作尝试去修改内存中的变量，如果失败则表示发生冲突，那么就应该有相应的重试逻辑。  
（2）悲观锁：还是像它的名字一样，对于并发间操作产生的线程安全问题持悲观状态，悲观锁认为竞争总是会发生，因此每次对某资源进行操作时，都会持有一个独占的锁，就像synchronized，不管三七二十一，直接上了锁就操作资源了。

## 171. Thread.sleep(0)的作用是什么？

【关键字：sleep】【难度：中级】【掌握程度：掌握】

解决方案：

由于Java采用抢占式的线程调度算法，因此可能会出现某条线程常常获取到CPU控制权的情况，为了让某些优先级比较低的线程也能获取到CPU控制权，可以使用Thread.sleep(0)手动触发一次操作系统分配时间片的操作，这也是平衡CPU控制权的一种操作。

## 172. 参数的值传递

【关键字：值传递】【难度：中级】【掌握程度：掌握】

解决方案：

基本类型和引用类型传递 实质上都是值传递，只不过基本类型传的是实际参数值副

## 173. CSS选择器的优先级是什么?

【关键字：CSS选择器】【难度：中级】【掌握程度：掌握】

解决方案：

从高到低依次是：  
1.在属性后面使用 !important会覆盖页面内任何位置定义的元素样式  
2.作为style属性写在元素标签上的内联样式  
3.id选择器  
4.类选择器  
5.伪类选择器  
6.属性选择器  
7.标签选择器  
8.通配符选择器  
9.浏览器自定义

## 174. 什么是Java内存模型？

【关键字：java内存】【难度：中级】【掌握程度：掌握】

解决方案：

Java内存模型定义了一种多线程访问Java内存的规范：  
（1）Java内存模型将内存分为了主内存和工作内存。类的状态，也就是类之间共享的变量，是存储在主内存中的，每次Java线程用到这些主内存中的变量的时候，会读一次主内存中的变量，并让这些内存在自己的工作内存中有一份拷贝，运行自己线程代码的时候，用到这些变量，操作的都是自己工作内存中的那一份。在线程代码执行完毕之后，会将最新的值更新到主内存中去  
（2）定义了几个原子操作，用于操作主内存和工作内存中的变量  
（3）定义了volatile变量的使用规则  
（4）happens-before，即先行发生原则，定义了操作A必然先行发生于操作B的一些规则，比如在同一个线程内控制流前面的代码一定先行发生于控制流后面的代码、一个释放锁unlock的动作一定先行发生于后面对于同一个锁进行锁定lock的动作等等，只要符合这些规则，则不需要额外做同步措施，如果某段代码不符合所有的happens-before规则，则这段代码一定是线程非安全的

## 175. 什么是CAS？

【关键字：java内存】【难度：中级】【掌握程度：掌握】

解决方案：

CAS，全称为Compare and Swap，即比较-替换。假设有三个操作数：内存值V、旧的预期值A、要修改的值B，当且仅当预期值A和内存值V相同时，才会将内存值修改为B并返回true，否则什么都不做并返回false。当然CAS一定要volatile变量配合，这样才能保证每次拿到的变量是主内存中最新的那个值，否则旧的预期值A对某条线程来说，永远是一个不会变的值A，只要某次CAS操作失败，永远都不可能成功。

## 176. css样式兼容不同浏览器问题？

【关键字：css样式】【难度：中级】【掌握程度：掌握】

解决方案：

在网站设计的时候，应该注意css样式兼容不同浏览器问题，特别是对完全使用DIV CSS设计的网，就应该更注意IE6 IE7 FF对CSS样式的兼容，不然，你的网乱可能出去不想出现的效果！具体讲了一些细节问题

## 177. Java中用到的线程调度算法是什么？

【关键字：线程调度】【难度：中级】【掌握程度：掌握】

解决方案：一个线程用完CPU之后，操作系统会根据线程优先级、线程饥饿情况等数据算出一个总的优先级并分配下一个时间片给某个线程执行。

## 178. Semaphore有什么作用？

【关键字：Semaphore】【难度：中级】【掌握程度：掌握】

解决方案：

Semaphore就是一个信号量，它的作用是限制某段代码块的并发数。Semaphore有一个构造函数，可以传入一个int型整数n，表示某段代码最多只有n个线程可以访问，如果超出了n，那么请等待，等到某个线程执行完毕这段代码块，下一个线程再进入。由此可以看出如果Semaphore构造函数中传入的int型整数n=1，相当于变成了一个synchronized了

## 179. 单例模式中的线程安全性是怎么样的？

【关键字：单例模式】【难度：初级】【掌握程度：掌握】

解决方案：

单例模式的线程安全意味着：某个类的实例在多线程环境下只会被创建一次出来。单例模式有很多种的写法：  
（1）饿汉式单例模式的写法：线程安全  
（2）懒汉式单例模式的写法：非线程安全  
（3）双检锁单例模式的写法：线程安全

## 180. 类的继承涉及方法重写、重载以及向上造型时候，对于属性的值的改变过程？

【关键字：继承】【难度：初级】【掌握程度：掌握】

解决方案：

写案例，画原理过程图

## 181. JS中this的指向问题？

【关键字：js】【难度：初级】【掌握程度：掌握】

解决方案：

this的指向在函数定义的时候是确定不了的，只有函数执行的时候才能确定this到底指向谁，实际上this的最终指向的是那个调用它的对象

## 182. 外部类和内部类同名变量，如何指定访问内部类变量名？

【关键字：内部类】【难度：初级】【掌握程度：掌握】

解决方案：

Outer.this.变量名

## 183. 可以通过this调用静态变量吗？

【关键字：静态变量】【难度：初级】【掌握程度：掌握】

解决方案：

可以，在实例方法中，可以通过this调用静态变量，相当于对象访问类属性  
而在静态方法里，是不可以

**183. 怎么唤醒一个阻塞的线程？**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：

如果线程是因为调用了wait()、sleep()或者join()方法而导致的阻塞，可以中断线程，并且通过抛出InterruptedException来唤醒它；如果线程遇到了IO阻塞，无能为力，因为IO是操作系统实现的，Java代码并没有办法直接接触到操作系统。

**184. 什么是多线程的上下文切换？**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：

多线程的上下文切换是指CPU控制权由一个已经正在运行的线程切换到另外一个就绪并等待获取CPU执行权的线程的过程

**185. 如果提交任务时线程池队列已满这时会发生什么？**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：

如果使用的是无界队列LinkedBlockingQueue，也就是无界队列的话，没关系，继续添加任务到阻塞队列中等待执行，因为LinkedBlockingQueue可以近乎认为是一个无穷大的队列，可以无限存放任务  
如果使用的是有界队列比如ArrayBlockingQueue，任务首先会被添加到ArrayBlockingQueue中，ArrayBlockingQueue满了，会根据maximumPoolSize的值增加线程数量，如果增加了线程数量还是处理不过来，ArrayBlockingQueue继续满，那么则会使用拒绝策略RejectedExecutionHandler处理满了的任务，默认是AbortPolicy

**186. 在做向上造型时，是否会调用父类的构造方法？**

【关键字：继承】【难度：初级】【掌握程度：掌握】

解决方案：

直接使用打桩输出，让学生很直观的看到确实调用了父类的无参构造器

**187. html里的id属性的值是唯一的，写代码的时候可以设置2个ID值一样的属性？**

【关键字：html】【难度：初级】【掌握程度：掌握】

解决方案：html代码不需要编译后运行，这样写也能被浏览器运行，但是在通过id属性获取元素的时候只能获取到第一个。

**188. 数组下标越界？**

【关键字：数组】【难度：初级】【掌握程度：掌握】

解决方案：在定义子弹的数组长度的时候，由于失误，写成了敌机的数组长度

## 189. 什么是AQS，有什么作用？

【关键字：流】【难度：初级】【掌握程度：掌握】

解决方案：

AQS全称为AbstractQueuedSychronizer，是抽象队列同步器  
AQS就是整个Java并发包的核心了，ReentrantLock、CountDownLatch、Semaphore等等都用到了它。AQS实际上以双向队列的形式连接所有的Entry，比方说ReentrantLock，所有等待的线程都被放在一个Entry中并连成双向队列，前面一个线程使用ReentrantLock好了，则双向队列实际上的第一个Entry开始运行。  
AQS定义了对双向队列所有的操作，而只开放了tryLock和tryRelease方法给开发者使用，开发者可以根据自己的实现重写tryLock和tryRelease方法，以实现自己的并发功能

**190. 类的继承涉及方法重写、重载以及向上造型时候，对于属性的值的改变过程**

【关键字：继承】【难度：初级】【掌握程度：掌握】

解决方案：

写案例，画原理过程图

**191. 同步监视器是什么？以及和同步锁之间的关系？**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：未解决

**192. 核心线程在没有资源请求的话处于什么样的状态**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：未解决

**193. 核心线程在没有资源请求的话处于什么样的状态**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：未解决

**194. 类的继承涉及方法重写、重载以及向上造型时候，对于属性的值的改变过程**

【关键字：继承】【难度：初级】【掌握程度：掌握】

解决方案：写案例，画原理过程图

**194. html时候用单引号，什么时候用双引号。**

【关键字：html】【难度：初级】【掌握程度：掌握】

解决方案：1、单引号和双引号都可以作为字符串的开始符和关闭符，并且只能‍同一种单或者双引号来定义开始和结束；单引号之间的字符都被认为是字符，即使是转义符\和变量符$,例外的是\’标识单引号。  
2.双引号之间的字符是需要php解析的，包括$,\，{都保留了php赋予的特殊含义。  
3." "双引号里面的字段会经过编译器解释，然后再当作HTML代码输出；而' '单引号里面的不进行解释，直接输出。

**195. jQuery选择器 table>tr 为什么不能选中table下的所有tr元素**

【关键字：jQuery】【难度：初级】【掌握程度：掌握】

解决方案：>是获取table下的直接子元素，table下的直接子元素是tbody。虽然代码里没写tbody但会被默认的加上。table>tbody>tr  这样可以获取到所有的tr元素

**196. Java中如何获取到线程dump文件**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：死循环、死锁、阻塞、页面打开慢等问题，打线程dump是最好的解决问题的途径。所谓线程dump也就是线程堆栈，获取到线程堆栈有两步：  
（1）获取到线程的pid，可以通过使用ps命令，在Linux环境下还可以使用ps -ef | grep java  
（2）打印线程堆栈，可以通过使用jstack pid命令，在Linux环境下还可以使用kill -3 pid  
另外提一点，Thread类提供了一个getStackTrace()方法也可以用于获取线程堆栈。这是一个实例方法，因此此方法是和具体线程实例绑定的，每次获取获取到的是具体某个线程当前运行的堆栈

**197. 生产者消费者模型的作用是什么**

【关键字：模型】【难度：初级】【掌握程度：掌握】

解决方案：（1）通过平衡生产者的生产能力和消费者的消费能力来提升整个系统的运行效率，这是生产者消费者模型最重要的作用  
（2）解耦，这是生产者消费者模型附带的作用，解耦意味着生产者和消费者之间的联系少，联系越少越可以独自发展而不需要受到相互的制约

**198. 怎么检测一个线程是否持有对象监视器**

【关键字：线程】【难度：初级】【掌握程度：掌握】

解决方案：可以判断某个线程是否持有对象监视器：Thread类提供了一个holdsLock(Object obj)方法，当且仅当对象obj的监视器被某条线程持有的时候才会返回true，注意这是一个static方法，这意味着"某条线程"指的是当前线程。

**199.**apImg=ImageIO.read(getClass().getResource("/imgs/airplane0.png"));//找不到图片

【关键字：IO流】【难度：初级】【掌握程度：掌握】

解决方案：检查.getResource("/imgs/airplane0.png"));路径名是否一致

**200. 没有敌机出来**

【关键字：】【难度：初级】【掌握程度：掌握】

### 解决方案：nextOne方法，把index定义到了方法里面，导致敌机赋值操作一直没有执行

**201. 子弹和小蜜蜂碰撞报空指针**

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：Hero类子弹生产的数组，doublefire应该是减2，学员写成-1了

**202. $(this) 和 this 关键字在 jQuery 中有什么区别**

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：$(this) 返回一个 jQuery 对象，可以对它调用多个 jQuery 方法，而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素

## 203. Tomcat的缺省是多少，怎么修改

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：Tomcat的缺省端口号是8080  
修改Tomcat端口号：  
1.找到Tomcat目录下的conf文件夹  
2.进入conf文件夹里面找到server.xml文件  
3.打开server.xml文件  
4.在server.xml文件里面找到下列信息  
maxThreads=”150″ minSpareThreads=”25″ maxSpareThreads=”75″  
enableLookups=”false” redirectPort=”8443″ acceptCount=”100″  
connectionTimeout=”20000″ disableUploadTimeout=”true” />  
5.把port=”8080″改成port=”8090″，并且保存  
6.启动Tomcat，并且在IE浏览器里面的地址栏输入<http://127.0.0.1:8090/>

## 204. The JAVA\_HOME environment variable is not defined This environment variable is needed to run this program；

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：没有在在tomcat的配置文件.bash\_profile中设置环境变量JAVA\_HOME，具体设置方法为：  
   JAVA\_HOME=/home/tomcat/j2sdk1.4.2\_08（具体值要以实际的jdk安装路径为准）  
   export JAVA\_HOMECLASSPATH=/home/tomcat/j2sdk1.4.2\_08/lib/tools.jar:/home/tomcat/j2sdk1.4.2\_08/lib/dt.jar  
   export CLASSPATH

## 205. tomcat 目录下的wtpwebapps里没有.class文件，eclipse启动tomcat之后，浏览器访问报500，代码都对。

【关键字：tomcat】【难度：初级】【掌握程度：掌握】

解决方案：停止tomcat 清除eclpise 缓存、tomcat目录缓存、浏览器缓存。再运行，tomcat 目录下的wtpwebapps里有.class文件，浏览器可以访问。

## 206. 项目按照正常步骤部署，始终访问不到，提示404错误。

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：tomcat的部署路径中有中文，将路径中的中文去掉之后就可以了。

## 207. cmd中导入jar包，cmd中编译servlet的时候提示找不到包。

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：jar的路径中有中文，不识别，需要将中文路径改掉。

## 208. eclipse启动tomcat，浏览器访问localhost无法加载页面

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：双击server窗口种的tomcat服务器，找到server locations，选择第二个Use Tomcat installation 保存

## 209. Servlet中很多类都报错，import也报错

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：没有导java EE的包，右键项目 properties--target runtimes 勾上tomcat 7.0 apply

## 210. tomcat如何解析https协议

【关键字：tomcat】【难度：初级】【掌握程度：掌握】

解决方案：1.生成安全证书命令：

 keytool -genkeypair -alias "tomcat" -keyalg "RSA" -keystore "f:\tomcat.keystore"   2.配置tomcat

## 211. ConcurrentHashMap的并发度

【关键字：hashMap】【难度：初级】【掌握程度：掌握】

解决方案：ConcurrentHashMap的并发度就是segment的大小，默认为16，这意味着最多同时可以有16条线程操作ConcurrentHashMap，这也是ConcurrentHashMap对Hashtable的最大优势，任何情况下，Hashtable不能同时有两条线程获取Hashtable中的数据

## 212. ReadWriteLock是什么

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：使用ReentrantLock，可能本身是为了防止线程A在写数据、线程B在读数据造成的数据不一致，但这样，如果线程C在读数据、线程D也在读数据，读数据是不会改变数据的，没有必要加锁，但是还是加锁了，降低了程序的性能。  
因为这个，才诞生了读写锁ReadWriteLock。ReadWriteLock是一个读写锁接口，ReentrantReadWriteLock是ReadWriteLock接口的一个具体实现，实现了读写的分离，读锁是共享的，写锁是独占的，读和读之间不会互斥，读和写、写和读、写和写之间才会互斥，提升了读写的性能.

## 212. 无法加载servlet类

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：jdk原先是用1.7后来改成1.8  jvm的版本跟不上jdk的版本导致的，只要把项目里面的java Compiler和Project Facets改成1.7就可以了

## 212. 用cmd控制台来加载servlet类失败

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：步骤弄混淆了，另外就是没有注意c盘和d盘的位置。按照正确的步骤操作就可以了

## 212. Linux环境下如何查找哪个线程使用CPU最长

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：（1）获取项目的pid，jps或者ps -ef | grep java，这个前面有讲过  
（2）top -H -p pid，顺序不能改变  
这样就可以打印出当前的项目，每条线程占用CPU时间的百分比。注意这里打出的是LWP，也就是操作系统原生线程的线程号。  
使用"top -H -p pid"+"jps pid"可以很容易地找到某条占用CPU高的线程的线程堆栈，从而定位占用CPU高的原因，一般是因为不当的代码操作导致了死循环。  
最后提一点，"top -H -p pid"打出来的LWP是十进制的，"jps pid"打出来的本地线程号是十六进制的，转换一下，就能定位到占用CPU高的线程的当前线程堆栈了.

## 212. 学员想知道输出集合的toString()方法在哪

【关键字：】【难度：初级】【掌握程度：掌握】

解决方案：直接找到ArrayList的父类的父类重写了Object的toString().

## 212. tomcat启动后，访问浏览器报500 ClassNotFoundException

【关键字：tomcat】【难度：初级】【掌握程度：掌握】

解决方案：500一般是代码出错，但是ClassNotFoundException是服务器没有把class文件部署成功。Tomcat上remove项目 clean服务器和project 再add项目到服务器中 重新publish。如果还不行，删除项目的jdk依赖重新添加。如果还不行，重启eclipse，重复上述操作。如果还不行，重装eclipse，重新部署tomcat。

## 213. 转发之后在页面出现乱码，但是控制台正常

【关键字：乱码】【难度：初级】【掌握程度：掌握】

解决方案：servlet创建的时候会自动生成一行response.getWriter()的代码，在这行代码之前必须要先处理乱码，否则服务器会默认iso8859-1

## 214. 为什么flush之后include还可以执行

【关键字：乱码】【难度：初级】【掌握程度：掌握】

解决方案：因为include是用于在响应中包含[其他](http://hi.baidu.com/skyland_lee/blog/item/:;" \t "_self)资源(Servlet，JSP页面或HTML文件)的内容。

   即请求转发后，原先的Servlet还可以继续输出响应信息，转发到的Servlet对请求做出的响应将并入原先Servlet的响应对象中。

## 215. Servlet的单例问题

【关键字：乱码】【难度：初级】【掌握程度：掌握】

解决方案：Servlet是一个供其他java程序调用的类，它不能独立运行，针对客户端的多次请求，通常状况下，Servlet只会创建一个Servlet实例对象，一旦创建它就会驻留在内存中，为后续的请求提供服务，直至退出web应用为止，也就是当我们关闭了浏览器之后我们的Servlet就终止了。  
当Servlet第一次访问的时候，就被加载到内存中，以后该实例对各个请求服务，没次情况会调用一次service方法。  
这样会出现什么问题：因为Servlet是单例的，所以会出现线程安全问题

## 216. incloud之前不设置响应编码，在包含进来的servlet里面设置的编码没有效果。

【关键字：乱码】【难度：初级】【掌握程度：掌握】

解决方案：incloud是将另一个servlet包含进来，在从servlet里面设置的编码对主servlet没有影响。

## 217. Linux在命令行中执行mkdir创建新文件夹时，可视化桌面目录不进行自动刷新，新建文件夹看不到

【关键字：乱码】【难度：初级】【掌握程度：掌握】

解决方案：可视化桌面系统在这个地方不会自动刷新，需要学员点击导航栏view,选择reload重新加载

## 218. 关于Servlet的生命周期，如何理解

【关键字：乱码】【难度：初级】【掌握程度：掌握】

解决方案：1.加载：容器通过类加载器使用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