**一： 选择题**

**1. 关于以下程序代码的说明正确的是（ D ）**

1．class HasStatic{

2． private static int x=100；

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

4． HasStatic hs1=new HasStatic( );

5． hs1.x++;

6． HasStatic hs2=new HasStatic( );

7． hs2.x++;

8． hs1=new HasStatic( );

9． hs1.x++;

10． HasStatic.x- -;

11． System.out.println(“x=”+x);

12． }

13． }

A、 5行不能通过编译，因为引用了私有静态变量

B、 10行不能通过编译，因为x是私有静态变量

C、 程序通过编译，输出结果为：x=103

D、 程序通过编译，输出结果为：x=102

**2. 下列关于for循环和while循环的说法中哪个是正确的？（ B ）**

A．while循环能实现的操作，for循环也都能实现

B．while循环判断条件一般是程序结果，for循环判断条件一般是非程序结果

C．两种循环任何时候都可替换

D．两种循环结构中都必须有循环体，循环体不能为空

**3. 以下选项中循环结构合法的是（ C ）**

A、while (int i<7)

{ i++;

System.out.println(“i is “+i);

}

B、 int j=3;

while(j)

{

System.out.println(“ j is “+j);

}

C、int j=0;

for(int k=0; j + k !=10; j++,k++)

{

System.out.println(“ j is “+ j + “k is”+ k);

}

D、 int j=0;

do{

System.out.println( “j is “+j++);

if (j = = 3) {continue loop;}

}while (j<10);

**4. 给出下面代码段, 哪行将引起一个编译时错误？（ D ）**

1) public class Test {

2) int n = 0;

3) int m = 0;

4) public Test(int a) { m=a; }

5) public static void main(String arg[]) {

6) Test t1,t2;

7) int j,k;

8) j=3; k=5;

9) t1=new Test();

10) t2=new Test(k);

11) }

12) }

A. 行1 B. 行4 C. 行6 D. 行9

**5. 下面的方法，当输入为2的时候返回值是多少?（ D ）**

public int getValue(int i) {

int result = 0;

switch (i) {

case 1:

result = result + i;

case 2:

result = result + i \* 2;

case 3:

result = result + i \* 3;

}

return result;

}

A 0 B 2 C 4 D 10

**二 ： 简述题**

**1、描述一下 JVM 加载 class 文件的原理机制?**

JVM 中类的装载是由 ClassLoader 和它的子类来实现的,Java ClassLoader 是一个重要的

Java 运行时系统组件。它负责在运行时查找和装入类文件的类。

**2**、**heap** 和 **stack** 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。

**3、GC 是什么?为什么要有 GC?**

GC 是垃圾收集的意思（Gabage Collection）,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。

**4、列举出一些常用的类，包，接口，请各举 5 个**

要让人家感觉你对 java ee 开发很熟，所以，不能仅仅只列 core java 中的那些东西，要多 列你在做 ssh 项目中涉及的那些东西。就写你最近写的那些程序中涉及的那些类。

常用的类：BufferedReader BufferedWriter FileReader FileWirter StringInteger java.util.Date，System，Class，List,HashMap

常用的包：java.lang java.io java.util java.sql,javax.servlet,org.apache.strtuts.action,org.hibernate 常用的接口：Remote List Map Document

NodeList,Servlet,HttpServletRequest,HttpServletResponse,Transaction(Hibernate)、

Session(Hibernate),HttpSession

**5 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？**

字节流，字符流。字节流继承于 InputStream OutputStream，字符流继承于 InputStream Reader

OutputStreamWriter。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。

**6、多线程有几种实现方法?同步有几种实现方法?**

多线程有两种实现方法，分别是继承 Thread 类与实现 Runnable 接口

同步的实现方面有两种，分别是 synchronized,wait 与 notify 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(中断异常)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。 Allnotity():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。

**7、说一说 Servlet 的生命周期?**

答:servlet 有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。 这个生存期由 javax.servlet.Servlet 接口的 init,service 和 destroy 方法表达。

Servlet 被服务器实例化后，容器运行其 init 方法，请求到达时运行其 service 方法，service 方法自动派遣运行与请求对应的 doXXX 方法（doGet，doPost）等，当服务器决定将实例 销毁的时候调用其 destroy 方法。

web 容器加载 servlet，生命周期开始。通过调用 servlet 的 init()方法进行 servlet 的初始化。 通过调用 service()方法实现，根据请求的不同调用不同的 do\*\*\*()方法。结束服务，web 容 器调用 servlet 的 destroy()方法。

**8、说出数据连接池的工作机制是什么?**

J2EE 服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端 程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空 闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池 连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。

实现方式，返回的 Connection 是原始 Connection 的代理，代理 Connection 的 close 方法 不是真正关连接，而是把它代理的 Connection 对象还回到连接池中。

**9、这段代码有什么不足之处?**

try {

Connection conn = ...; Statement stmt = ...;

ResultSet rs =stmt.executeQuery("select \* from table1");

while(rs.next()) {

}

} catch(Exception ex) {

}

答：没有 finally 语句来关闭各个对象，另外，使用 finally 之后，要把变量的定义放在 try 语 句块的外面，以便在 try 语句块之外的 finally 块中仍可以访问这些变量。

**10、一个用户具有多个角色，****请查询出该表中具有该用户的所有角色的其他用户。**

select count(\*) as num,tb.id from

tb,

(select role from tbwhere id=xxx) as t1 where

tb.role = t1.role and tb.id != t1.id

group by tb.id having

num = select count(role)from tb where id=xxx;

**11、一个用户表中有一个积分字段，假如数据库中有 100 多万个用户，若要在 每年第一天凌晨将积分清零，你将考虑什么，你将想什么办法解决?**

alter table drop column score;

alter table add colunm score int; 可能会很快，但是需要试验，试验不能拿真实的环境来操刀，并且要注意， 这样的操作时无法回滚的，在我的印象中，只有 inert update delete 等 DML 语句才能回滚， 对于 create table,drop table ,alter table 等 DDL 语句是不能回滚。

解决方案一，update user set score=0; 解决方案二，假设上面的代码要执行好长时间，超出我们的容忍范围，那我就 alter table user drop columnscore;alter table user add column score int。

下面代码实现每年的那个凌晨时刻进行清零。

Runnable runnable =

new Runnable(){

public voidrun(){

clearDb();

schedule(this,newDate(new Date().getYear()+1,0,0));

}

};

schedule(runnable,

new Date(newDate().getYear()+1,0,1));

**12、你对 Spring 的理解, 什么是 spring 的 IOC AOP。**

1.Spring 实现了工厂模式的工厂类（在这里有必要解释清楚什么是工厂模式），这个类名为 BeanFactory（实际上是一个接口），在程序中通常 BeanFactory 的子类 ApplicationContext。 Spring 相当于一个大的工厂类，在其配置文件中通过<bean>元素配置用于创建实例对象的 类名和实例对象的属性。

2. Spring 提供了对 IOC 良好支持，IOC 是一种编程思想，是一种架构艺术，利用这种思想可以很好地实现模块之间的解耦。IOC 也称为 D（I

Depency Injection），什么叫依赖注入呢？

譬如，Class Programmer

{

Computer computer =null;

public void code()

{

//Computercomputer = new IBMComputer();

//Computercomputer = beanfacotry.getComputer();

computer.write();

}

public voidsetComputer(Computer computer)

{

this.computer= computer;

}

} 另外两种方式都由依赖，第一个直接依赖于目标类，第二个把依赖转移到工厂上，第三个彻 底与目标和工厂解耦了。在 spring 的配置文件中配置片段如下：

<bean id=”computer” class=”cn.itcast.interview.Computer”>

</bean>

<bean id=”programmer” class=”cn.itcast.interview.Programmer”>

<property name=”computer” ref=”computer”></property>

</bean>

3. Spring 提供了对 AOP技术的良好封装， AOP 称为面向切面编程，就是系统中有很多各不相干的类的方法，在这些众多方法中要加入某种系统功能的代码，例如，加入日志，加入权限判断，加入异常处理，这种应用称为 AOP。实现 AOP 功能采用的是代理技术，客户端 程序不再调用目标，而调用代理类，代理类与目标类对外具有相同的方法声明，有两种方式 可以实现相同的方法声明，一是实现相同的接口，二是作为目标的子类在，JDK 中采用 Proxy 类产生动态代理的方式为某个接口生成实现类，如果要为某个类生成子类，则可以用 CGLI B。在生成的代理类的方法中加入系统功能和调用目标类的相应方法，系统功能的代理以 Advice 对象进行提供，显然要创建出代理对象，至少需要目标类和 Advice 类。spring 提供 了这种支持，只需要在 spring 配置文件中配置这两个元素即可实现代理和 aop 功能，例如，

<bean id=”proxy” type=”org.spring.framework.aop.ProxyBeanFactory”>

<property name=”target”ref=””></property>

<property name=”advisor”ref=””></property>

</bean>