1.Overload和Override的区别。Overloaded的方法是否可以改变返回值的类型?

方法的重写Overriding和重载Overloading是Java多态性的不同表现。

重写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。

如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重(Overriding)。

子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被“屏蔽”了。

如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded的方法是可以改变返回值的类型。

2.servlet的生命周期

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。

servlet生命周期.jpg 见附件

3.说出ArrayList,Vector, LinkedList的存储性能和特性

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，如果元素的数目超出了内部数组目前的长度它们都需要扩展内部数组的长度，Vector缺省情况下自动增长原来一倍的数组长度，ArrayList是原来的50%，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同步方法保证线程安全，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。

4.final, finally, finalize的区别。

　　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。

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，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

5.forward 和redirect的区别

forward是服务器请求资源，服务器直接访问目标地址的URL，把那个URL的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

redirect就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request参数都可以获取。

6.JSP中动态INCLUDE与静态INCLUDE的区别？

动态INCLUDE用jsp:include动作实现 <jsp:include page="included.jsp" flush="true" />它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数。

静态INCLUDE用include伪码实现,不会检查所含文件的变化，适用于包含静态页面<%@ include file="included.htm" %>

7.设计4个线程，其中两个线程每次对j增加1，另外两个线程对j每次减少1。写出程序。

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 ThreadTest1{

private int j;

public static void main(String args[]){

ThreadTest1 tt=new ThreadTest1();

Inc inc=tt.new Inc();

Dec dec=tt.new Dec();

for(int i=0;i<2;i++){

Thread t=new Thread(inc);

t.start();

t=new Thread(dec);

t.start();

}

}

private synchronized void inc(){

j++;

System.out.println(Thread.currentThread().getName()+"-inc:"+j);

}

private synchronized void dec(){

j--;

System.out.println(Thread.currentThread().getName()+"-dec:"+j);

}

class Inc implements Runnable{

public void run(){

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

inc();

}

}

}

class Dec implements Runnable{

public void run(){

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

dec();

}

}

}

}

8.编程题: 写一个Singleton出来。

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。

一般Singleton模式通常有几种种形式:

第一种形式: 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。

public class Singleton {

private Singleton(){}

　　 //在自己内部定义自己一个实例，是不是很奇怪？

　　 //注意这是private 只供内部调用

　　 private static Singleton instance = new Singleton();

　　 //这里提供了一个供外部访问本class的静态方法，可以直接访问

　　 public static Singleton getInstance() {

　　　　 return instance;

　　 }

}

第二种形式:

public class Singleton {

　　private static Singleton instance = null;

　　public static synchronized Singleton getInstance() {

　　//这个方法比上面有所改进，不用每次都进行生成对象，只是第一次

　　//使用时生成实例，提高了效率！

　　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

9.编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串。 但是要保证汉字不被截半个，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”。

Java代码

1.public class SubStringTest {

2.

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

4. String inStr = "我ABC";

5. String str = subString(inStr, 4);

6. System.out.println(str);

7.

8. inStr = "我ABC汉DEF";

9. str = subString(inStr, 6);

10. System.out.println(str);

11. }

12.

13. /\*\*

14. \* 字符串按字节截取

15. \* @param str 原字符

16. \* @param len 截取长度

17. \* @return String

18. \* @author zheng

19. \*/

20. public static String subString(String str, int len) {

21. if (str == null || ("").equals(str)) {

22. return "";

23. }

24. byte[] strByte = null;

25. try {

26. strByte = str.getBytes("GBK");

27. } catch (UnsupportedEncodingException e) {

28. e.printStackTrace();

29. }

30. int strLen = strByte.length;

31. if (len >= strLen || len < 1) {

32. return str;

33. }

34. int count = 0;

35. for (int i = 0; i < len; i++) {

36. int value = (int) strByte[i];

37. if (value < 0) {

38. count++;

39. }

40. }

41. if (count % 2 != 0) {

42. len = (len == 1) ? len + 1 : len - 1;

43. }

44.

45. return str.substring(0, len-1);

46. }

47.}

8.编程题: 写一个Singleton出来。

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。

一般Singleton模式通常有几种种形式:

第一种形式: 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。

public class Singleton {

private Singleton(){}

　　 //在自己内部定义自己一个实例，是不是很奇怪？

　　 //注意这是private 只供内部调用

　　 private static Singleton instance = new Singleton();

　　 //这里提供了一个供外部访问本class的静态方法，可以直接访问

　　 public static Singleton getInstance() {

　　　　 return instance;

　　 }

}

第二种形式:

public class Singleton {

　　private static Singleton instance = null;

　　public static synchronized Singleton getInstance() {

　　//这个方法比上面有所改进，不用每次都进行生成对象，只是第一次

　　//使用时生成实例，提高了效率！

　　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

这个好像会这么写：

public class SingletonClass{

private static class ResourceHolder{

public Instance instance=new Instance();

}

public Instance getInstance(){

return SingletonClass.ResouceHolder.instance;

}

public static class Instance{

}

}

}

或者用枚举实现吧。