代码与编程题

135、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

136、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？

答:父类:

```
package test;
public class  FatherClass
{
    public FatherClass()
 {
  System.out.println("FatherClass Create");
 }
}
```

子类:

```
package test;
import test.FatherClass;
public class  ChildClass extends FatherClass
{
 public ChildClass()
 {
```

```
  System.out.println("ChildClass Create");
 }
 public static void main(String[] args)
 {
  FatherClass fc = new FatherClass();
  ChildClass cc = new ChildClass();
 }
}
```

输出结果：

```
C:\>java test.ChildClass
FatherClass Create
FatherClass Create
ChildClass Create
```

137、内部类的实现方式?

答:示例代码如下：

```
package test;
public class  OuterClass
{
 private class InterClass
 {
  public InterClass()
  {
   System.out.println("InterClass Create");
  }
 }
 public OuterClass()
 {
  InterClass ic = new InterClass();
  System.out.println("OuterClass Create");
 }
 public static void main(String[] args)
 {
  OuterClass oc = new OuterClass();
 }
}
```

输出结果:

```
C:\>java test/OuterClass
InterClass Create
OuterClass Create
```

再一个例题：

```
public class OuterClass {
  private double d1 = 1.0;
    //insert code here
}
```

You need to insert an inner class declaration at line 3. Which two inner class declarations are

valid?(Choose two.)
A. class InnerOne{
    public static double methoda() {return d1;}
  }
B. public class InnerOne{
    static double methoda() {return d1;}
  }
C. private class InnerOne{
    double methoda() {return d1;}
  }
D. static class InnerOne{
    protected double methoda() {return d1;}
  }
E. abstract class InnerOne{
    public abstract double methoda();
  }
说明如下:
一.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
二.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;return d1 出错。

故 D 错
三.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
四.答案为C、E

138、Java 的通信编程,编程题(或问答),用JAVA SOCKET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 Socket socket;
 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("A client come in!IP:"+RemoteIP+RemotePort);
    in = new BufferedReader(new

InputStreamReader(socket.getInputStream()));
    String line = in.readLine();
    System.out.println("Cleint send is :" + line);
    out = new PrintWriter(socket.getOutputStream(),true);
    out.println("Your Message Received!");
    out.close();
    in.close();
    socket.close();
   }
  }catch (IOException e)
  {
   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("Try to Connect to 127.0.0.1:10000");
   socket = new Socket("127.0.0.1",10000);
   System.out.println("The Server Connected!");
   System.out.println("Please enter some Character:");
   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(IOException e)
  {
   out.println("Wrong");
  }
 }
 public static void main(String[] args)
 {
  new Client();
 }
};
```

139、用JAVA实现一种排序，JAVA类实现序列化的方法(二种)？
如在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("The ArrayList Sort Before:");
  for (int i=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()
 {
  Integer tempInt;
  int MaxSize=1;
  for(int i=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 (int j=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("The ArrayList Sort After:");
  for(int i=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();
 }
}
```

140、编程: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"。
答:代码如下:

```
package test;

class  SplitString
{
 String SplitStr;
 int SplitByte;
 public SplitString(String str,int bytes)
 {
  SplitStr=str;
  SplitByte=bytes;
  System.out.println("The String is:'"+SplitStr+"';SplitBytes="+SplitByte);
 }
 public void SplitIt()
 {
```

```
  int loopCount;


loopCount=(SplitStr.length()%SplitByte==0)?(SplitStr.length()/SplitByte):(SplitStr.length()/Split

Byte+1);
  System.out.println("Will Split into "+loopCount);
  for (int i=1;i<=loopCount ;i++ )
  {
   if (i==loopCount){


System.out.println(SplitStr.substring((i-1)*SplitByte,SplitStr.length()));
   } else {


System.out.println(SplitStr.substring((i-1)*SplitByte,(i*SplitByte)));
   }
  }
 }
 public static void main(String[] args)
 {
  SplitString ss = new SplitString("test中dd文dsaf中男大3443n中国43中国人

0ewldfls=103",4);
  ss.SplitIt();
 }
}
```

141、JAVA多线程编程。用JAVA写一个多线程程序，如写四个线程，二个加1，二个对一个变量减一，输出。
希望大家补上，谢谢

142、可能会让你写一段Jdbc连Oracle的程序,并实现数据查询.
答:程序如下：

```
package hello.ant;
import java.sql.*;
public class  jdbc
{
 String dbUrl="jdbc:oracle:thin:@127.0.0.1:1521:orcl";
 String theUser="admin";
 String thePw="manager";
 Connection c=null;
 Statement conn;
 ResultSet rs=null;
 public jdbc()
 {
  try{
```

```
   Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
        c = DriverManager.getConnection(dbUrl,theUser,thePw);
   conn=c.createStatement();
 }catch(Exception e){
  e.printStackTrace();
 }
}
public boolean executeUpdate(String sql)
{
  try
  {
    conn.executeUpdate(sql);
    return true;
  }
  catch (SQLException e)
  {
    e.printStackTrace();
    return false;
  }
}
public ResultSet executeQuery(String sql)
{
  rs=null;
  try
  {
    rs=conn.executeQuery(sql);
  }
  catch (SQLException e)
  {
    e.printStackTrace();
  }
  return rs;
}
public void close()
{
  try
  {
    conn.close();
    c.close();
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
}
public static void main(String[] args)
```

```
 {
  ResultSet rs;
  jdbc conn = new jdbc();
  rs=conn.executeQuery("select * from test");
  try{
  while (rs.next())
  {
   System.out.println(rs.getString("id"));
   System.out.println(rs.getString("name"));
  }
  }catch(Exception e)
  {
   e.printStackTrace();
  }
 }
}
```

143、ORACLE大数据量下的分页解决方法。一般用截取ID方法，还有是三层嵌套方法。

答:一种分页方法

```
<%
  int i=1;
  int numPages=14;
  String pages = request.getParameter("page") ;
  int currentPage = 1;
  currentPage=(pages==null)?(1):{Integer.parseInt(pages)}
  sql = "select count(*) from tables";
  ResultSet rs = DBLink.executeQuery(sql) ;
  while(rs.next()) i = rs.getInt(1) ;
  int intPageCount=1;
  intPageCount=(i%numPages==0)?(i/numPages):(i/numPages+1);
  int nextPage ;
  int upPage;
  nextPage = currentPage+1;
  if (nextPage>=intPageCount) nextPage=intPageCount;
  upPage = currentPage-1;
  if (upPage<=1) upPage=1;
  rs.close();
  sql="select * from tables";
  rs=DBLink.executeQuery(sql);
  i=0;
  while((i<numPages*(currentPage-1))&&rs.next()){i++;}
%>
//输出内容
//输出翻页连接
合计:<%=currentPage%>/<%=intPageCount%><a href="List.jsp?page=1">第一页</a><a
```

```
href="List.jsp?page=<%=upPage%>">上一页</a>
<%
  for(int j=1;j<=intPageCount;j++){
  if(currentPage!=j){
%>
  <a href="list.jsp?page=<%=j%>">[<%=j%>]</a>
<%
  }else{
  out.println(j);
  }
  }
%>
<a href="List.jsp?page=<%=nextPage%>">下一页</a><a
href="List.jsp?page=<%=intPageCount%>">最后页

</a>
```

144、用jdom解析xml文件时如何解决中文问题?如何解析?

答:看如下代码,用编码方式加以解决

```
package test;
import java.io.*;
public class DOMTest
{
 private String inFile = "c:\\people.xml";
 private String outFile = "c:\\people.xml";
 public static void main(String args[])
 {
    new DOMTest();
   }
 public DOMTest()
 {
  try
    {
     javax.xml.parsers.DocumentBuilder builder =


javax.xml.parsers.DocumentBuilderFactory.newInstance().newDocumentBuilder();
      org.w3c.dom.Document doc = builder.newDocument();
      org.w3c.dom.Element root = doc.createElement("老师");
      org.w3c.dom.Element wang = doc.createElement("王");
   org.w3c.dom.Element liu = doc.createElement("刘");
      wang.appendChild(doc.createTextNode("我是王老师"));
      root.appendChild(wang);
      doc.appendChild(root);
      javax.xml.transform.Transformer transformer =
       javax.xml.transform.TransformerFactory.newInstance().newTransformer();
```

```
      transformer.setOutputProperty(javax.xml.transform.OutputKeys.ENCODING, "gb2312");
      transformer.setOutputProperty(javax.xml.transform.OutputKeys.INDENT, "yes");


      transformer.transform(new javax.xml.transform.dom.DOMSource(doc),
           new

javax.xml.transform.stream.StreamResult(outFile));
     }
     catch (Exception e)
     {
      System.out.println (e.getMessage());
     }
    }
}
```

145、编程用JAVA解析XML的方式.

答:用SAX方式解析XML，XML文件如下：

```
<?xml version="1.0" encoding="gb2312"?>
<person>
  <name>王小明</name>
  <college>信息学院</college>
  <telephone>6258113</telephone>
  <notes>男,1955年生,博士，95年调入海南大学</notes>
 </person>
```

事件回调类SAXHandler.java

```
 import java.io.*;
import java.util.Hashtable;
import org.xml.sax.*;
public class SAXHandler extends HandlerBase
  {
  private Hashtable table = new Hashtable();
  private String currentElement = null;
  private String currentValue = null;
  public void setTable(Hashtable table)
    {
    this.table = table;
    }
  public Hashtable getTable()
    {
    return table;
    }
  public void startElement(String tag, AttributeList attrs)
  throws SAXException
    {
    currentElement = tag;
```

```
  }
 public void characters(char[] ch, int start, int length)
 throws SAXException
  {
  currentValue = new String(ch, start, length);
  }
 public void endElement(String name) throws SAXException
  {
  if (currentElement.equals(name))
    table.put(currentElement, currentValue);
  }
 }
```

JSP内容显示源码,SaxXml.jsp:

```
<HTML>
<HEAD>
<TITLE>剖析XML文件people.xml</TITLE>
</HEAD>
<BODY>
<%@ page errorPage="ErrPage.jsp"
contentType="text/html;charset=GB2312" %>
<%@ page import="java.io.*" %>
<%@ page import="java.util.Hashtable" %>
<%@ page import="org.w3c.dom.*" %>
<%@ page import="org.xml.sax.*" %>
<%@ page import="javax.xml.parsers.SAXParserFactory" %>
<%@ page import="javax.xml.parsers.SAXParser" %>
<%@ page import="SAXHandler" %>
<%
File file = new File("c:\\people.xml");
FileReader reader = new FileReader(file);
Parser parser;
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();
SAXHandler handler = new SAXHandler();
sp.parse(new InputSource(reader), handler);
Hashtable hashTable = handler.getTable();
out.println("<TABLE BORDER=2><CAPTION>教师信息表</CAPTION>");
out.println("<TR><TD>姓名</TD>" + "<TD>" +
  (String)hashTable.get(new String("name")) + "</TD></TR>");
out.println("<TR><TD>学院</TD>" + "<TD>" +
  (String)hashTable.get(new String("college"))+"</TD></TR>");
out.println("<TR><TD>电话</TD>" + "<TD>" +
  (String)hashTable.get(new String("telephone")) + "</TD></TR>");
out.println("<TR><TD>备注</TD>" + "<TD>" +
  (String)hashTable.get(new String("notes")) + "</TD></TR>");
out.println("</TABLE>");
```

```
%>
</BODY>
</HTML>
```

146、EJB的基本架构

答:一个EJB包括三个部分:

Remote Interface 接口的代码

```
package Beans;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
public interface Add extends EJBObject
{
 //some method declare
}
```

Home Interface 接口的代码

```
package Beans;
import java.rmi.RemoteException;
import jaax.ejb.CreateException;
import javax.ejb.EJBHome;
public interface AddHome extends EJBHome
{
  //some method declare
}
```

EJB类的代码

```
package Beans;
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javx.ejb.SessionContext;
public class AddBean Implements SessionBean
{
  //some method declare
}
```

147、如何校验数字型?

```
var re=/^\d{1,8}$|\.\d{1,2}$/;
var str=document.form1.all(i).value;
var r=str.match(re);
if (r==null)
{
  sign=-4;
  break;
}
else{
  document.form1.all(i).value=parseFloat(str);
}
```

148、将一个键盘输入的数字转化成中文输出
(例如:输入:1234567　　输出:一百二拾三万四千五百六拾七)
用java语言实现,,请编一段程序实现!

```
public class Reader {
  private String strNum;
  private String strNumChFormat;
  private String strNumTemp;
  private int intNumLen;
  private String strBegin;
  public Reader(String strNum) {
    this.strNum = strNum;
  }
  public boolean check(String strNum) {
    boolean valid = false;

    if (strNum.substring(0,1).equals("0")){
     this.strNum = strNum.substring(1);
    }
    try {
      new Double(strNum);
      valid = true;
    }
    catch (NumberFormatException ex) {
      System.out.println("Bad number format!");
    }
    return valid;
  }
  public void init() {
    strNumChFormat = "";
    intNumLen = strNum.length();
    strNumTemp = strNum;
    strNumTemp = strNumTemp.replace('1', '一');
    strNumTemp = strNumTemp.replace('2', '二');
    strNumTemp = strNumTemp.replace('3', '三');
    strNumTemp = strNumTemp.replace('4', '四');
    strNumTemp = strNumTemp.replace('5', '五');
    strNumTemp = strNumTemp.replace('6', '六');
    strNumTemp = strNumTemp.replace('7', '七');
    strNumTemp = strNumTemp.replace('8', '八');
    strNumTemp = strNumTemp.replace('9', '九');
    strNumTemp = strNumTemp.replace('0', '零');
    strNumTemp = strNumTemp.replace('.', '点');
    strBegin = strNumTemp.substring(0, 1);
  }
  public String readNum() {
    if (check(strNum)) {
```

```
init();
try {
  for (int i = 1, j = 1, k = 1; i < intNumLen; i++) {
    if (strNumTemp.charAt(intNumLen - 1) == '零' && i == 1) {
      strNumChFormat = "位";
    }
    else if (strNumTemp.charAt(intNumLen - i) == '零' && j == 1) {
      strNumChFormat = "位" + strNumChFormat;
    }
    else if (strNumTemp.charAt(intNumLen - i) == '点') {
      j = 1;
      k = 1;
      strNumChFormat = strNumTemp.charAt(intNumLen - i) + strNumChFormat;
      continue;
    }
    else {
      strNumChFormat = strNumTemp.charAt(intNumLen - i) + strNumChFormat;
    }
    if (strNumTemp.charAt(intNumLen - i - 1) != '位' &&
        strNumTemp.charAt(intNumLen - i - 1) != '零') {
      if (j == 1 && i < intNumLen) {
        strNumChFormat = '拾' + strNumChFormat;
      }
      else if (j == 2 && i < intNumLen) {
        strNumChFormat = '百' + strNumChFormat;
      }
      else if (j == 3 && i < intNumLen) {
        strNumChFormat = '千' + strNumChFormat;
      }
    }
    if (j == 4 && i < intNumLen) {
      j = 0;
    }
    if (k == 4 && i < intNumLen) {
      strNumChFormat = '万' + strNumChFormat;
    }
    else if (k == 8 && i < intNumLen) {
      k = 0;
      strNumChFormat = '亿' + strNumChFormat;
    }
    j++;
    k++;
  }
  while (strNumChFormat.indexOf("位") != -1) {
    strNumChFormat = strNumChFormat.replaceAll("位", " ");
  }
```

```
        if (strNumChFormat.substring(0, 2) == "一拾") {
          strNumChFormat = strNumChFormat.substring(1, strNumChFormat.length());
        }
        if (strNumChFormat.indexOf("点") >= 0) {
          String rebegin = strNumChFormat.substring(0,
              strNumChFormat.indexOf("点"));
          String relast = strNumChFormat.substring(strNumChFormat.indexOf("点"),
              strNumChFormat.length());
          for (int i = 1; i <= relast.length(); i++) {
            relast = relast.replaceAll("拾", "");
            relast = relast.replaceAll("百", "");
            relast = relast.replaceAll("千", "");
            relast = relast.replaceAll("万", "");
            relast = relast.replaceAll("亿", "");
          }
          strNumChFormat = rebegin + relast;
        }
      }
      catch (ArrayIndexOutOfBoundsException ex) {
        ex.printStackTrace();
      }
      catch (Exception ex) {
        ex.printStackTrace();
      }
      int off = strNumChFormat.indexOf("点");
      strNumChFormat = strBegin + strNumChFormat.substring(0);
    }
    else {
      strNumChFormat = "";
    }
    return strNumChFormat;
  }
  public static void main(String args[]) {
    try {
      String number = args[0].toString();
      System.out.println("The number is: " + number);
      Reader reader = new Reader(number);
      System.out.println("Output String: " + reader.readNum());
    }
    catch (Exception ex) {
      System.out.println("Please input like that: javac Reader <number>");
    }
  }
}
```

149、JAVA代码查错

1.

```
abstract class Name {
  private String name;
  public abstract boolean isStupidName(String name) {}
}
```

大侠们，这有何错误?

答案: 错。abstract method必须以分号结尾，且不带花括号。

2.

```
public class Something {
  void doSomething () {
    private String s = "";
    int l = s.length();
  }
}
```

有错吗?

答案: 错。局部变量前不能放置任何访问修饰符 (private，public，和protected)。final可以用来修饰局部变量(final如同abstract和strictfp，都是非访问修饰符，strictfp只能修饰class和method而非variable)。

3.

```
abstract class Something {
  private abstract String doSomething ();
}
```

这好像没什么错吧?

答案: 错。abstract的methods不能以private修饰。abstract的methods就是让子类implement(实现)具体细节的，怎么可以用private把abstract
method封锁起来呢? (同理，abstract method前不能加final)。

4.

```
public class Something {
  public int addOne(final int x) {
    return ++x;
  }
}
```

这个比较明显。

答案: 错。int x被修饰成final，意味着x不能在addOne method中被修改。

5.

```
public class Something {
  public static void main(String[] args) {
    Other o = new Other();
    new Something().addOne(o);
  }
  public void addOne(final Other o) {
    o.i++;
  }
}
class Other {
  public int i;
}
```

和上面的很相似，都是关于final的问题，这有错吗?

答案: 正确。在addOne method中，参数o被修饰成final。如果在addOne method里我们修改了o的reference (比如: o = new Other();)，那么如同上例这题也是错的。但这里修改的是o的member vairable (成员变量)，而o的reference并没有改变。

6.

```
class Something {
    int i;
    public void doSomething() {
        System.out.println("i = " + i);
    }
}
```

有什么错呢? 看不出来啊。

答案: 正确。输出的是"i = 0"。int i属於instant variable (实例变量，或叫成员变量)。instant variable有default value。int的default value是0。

7.

```
class Something {
    final int i;
    public void doSomething() {
        System.out.println("i = " + i);
    }
}
```

和上面一题只有一个地方不同，就是多了一个final。这难道就错了吗?

答案: 错。final int i是个final的instant variable (实例变量，或叫成员变量)。final的instant variable没有default value，必须在constructor (构造器)结束之前被赋予一个明确的值。可以修改为"final int i = 0;"。

8.

```
public class Something {
     public static void main(String[] args) {
        Something s = new Something();
        System.out.println("s.doSomething() returns " + doSomething());
    }
    public String doSomething() {
        return "Do something ...";
    }
}
```

看上去很完美。

答案: 错。看上去在main里call doSomething没有什么问题，毕竟两个methods都在同一个class里。但仔细看，main是static的。static method不能直接call non-static methods。可改成"System.out.println("s.doSomething() returns " + s.doSomething());"。同理，static method不能访问non-static instant variable。

9.

此处，Something类的文件名叫OtherThing.java

```
class Something {
    private static void main(String[] something_to_do) {
        System.out.println("Do something ...");
    }
}
```

这个好像很明显。

答案: 正确。从来没有人说过Java的Class名字必须和其文件名相同。但public class的名字必须和文件名相同。

10.

```
interface  A{
  int x = 0;
}
class B{
  int x =1;
}
class C extends B implements A {
  public void pX(){
    System.out.println(x);
  }
  public static void main(String[] args) {
    new C().pX();
  }
}
```

答案:错误。在编译时会发生错误(错误描述不同的JVM有不同的信息，意思就是未明确的x调用，两个x都匹配(就象在同时import java.util和java.sql两个包时直接声明Date一样)。对于父类的变量,可以用super.x来明确，而接口的属性默认隐含为 public static final.所以可以通过A.x来明确。

11.

```
interface Playable {
   void play();
}
interface Bounceable {
   void play();
}
interface Rollable extends Playable, Bounceable {
   Ball ball = new Ball("PingPang");
}
class Ball implements Rollable {
   private String name;
   public String getName() {
      return name;
   }
   public Ball(String name) {
      this.name = name;
   }
  public void play() {
      ball = new Ball("Football");
      System.out.println(ball.getName());
   }
}
```

这个错误不容易发现。

答案: 错。"interface Rollable extends Playable, Bounceable"没有问题。interface可继承多个interfaces，所以这里没错。问题出在interface Rollable里的"Ball ball = new Ball("PingPang");"。任何在interface里声明的interface variable (接口变量，也可称成员变量)，默认为public static final。也就是说"Ball ball = new Ball("PingPang");"实际上是"public static final

Ball ball = new Ball("PingPang");"。在Ball类的Play()方法中，"ball = new Ball("Football");"改变了ball的reference，而这里的ball来自Rollable interface，Rollable interface里的ball是public static final的，final的object是不能被改变reference的。因此编译器将在"ball = new Ball("Football");"这里显示有错。

28、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();
}
 }
  }
}
```