面试题集

面试题集共分为以下十部分：

一、CoreJava 部分：（共 95 题：基础 91 道，中等难度 4 道）

基础及语法部分：（共 61 题：基础 60 道、中等难度 1 道）

1、面向对象的特征有哪些方面？【基础】

答：面向对象的特征主要有以下几个方面：

1)抽象：抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。抽象包括两个方面，一是过程抽象,二是数据抽象。

2)继承：继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。

3)封装：封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象,这些对象通过一个受保护的接口访问其他对象。

4)多态性：多态性是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

2、作用域 public,private,protected,以及不写时的区别？【基础】

答：区别如下：

作用域	当前类	同包	子孙类	其他
public	√	√	√	√

protected	√	√	√	×
default	√	√	×	×
private	√	×	×	×

不写时默认为 default。

3、String 是最基本的数据类型吗? 【基础】
答：不是。

4、float 型 float f=3.4是否正确? 【基础】
答:不正确;精度不准确,应该用强制类型转换,如下所示:float f=(float)3.4 。

5、语句 float f=1.3; 编译能否通过? 【基础】
答:不能; 应该用强制类型转换，如下所示：float f=(float)1.3; 。

6、short s1 = 1; s1 = s1 + 1;有什么错?
short s1 = 1; s1 += 1;有什么错? 【基础】
答：short s1 = 1; s1 = s1 + 1;s1+1运算结果是 int 型，需要强制转换类型;
short s1 = 1; s1 += 1;可以正确编译,自动类型提升。

7、Java 有没有 goto? 【基础】
答：goto 是 java 中的保留字，现在没有在 java 中使用。

8、int 和 Integer 有什么区别? 【基础】
答：Java 提供两种不同的类型：引用类型和原始类型（或内置类型）;
int 是 java 的原始数据类型，Integer 是 java 为 int 提供的封装类。
Java 为每个原始类型提供了封装类：
原始类型: boolean,char,byte,short,int,long,float,double
封装类型：Boolean，Character，Byte，Short，Integer，Long，Float，Double
引用类型和原始类型的行为完全不同，并且它们具有不同的语义。引用类型和原始类型具有不同的特征和用法，它们包括：大小和速度问题，这种类型以哪种类型的数据结构存储，当引用类型和原始类型用作某个类的实例数据时所指定的缺省值。对象引用实例变量的缺省值为 null，而原始类型实例变量的缺省值与它们的类型有关。

9、&和&&的区别? 【基础】
答：&是位运算符，表示按位与运算，&&是逻辑运算符，表示逻辑与（and）。

10、简述逻辑操作(&,|,^)与条件操作(&&,||)的区别? 【基础】
答：区别主要有两点：a.条件操作只能操作布尔型的,而逻辑操作不仅可以操作布尔型,而且可以操作数值型 b.逻辑操作不会产生短路。

11、heap 和 stack 有什么区别? 【基础】
答：栈是一种线形集合，其添加和删除元素的操作应在同一段完成，栈按照后进先出的方式进行处理；堆是栈的一个组成元素。

12、Math.round(11.5) 等于多少? Math.round(-11.5)等于多少? 【基础】
答：Math.round(11.5)==12　Math.round(-11.5)==-11　round 方法返回与参数最接近的长整数，参数加 1/2 后求其 floor。

13、swtich 是否能作用在 byte 上，是否能作用在 long 上，是否能作用在 String 上? 【基础】
答：switch（expr1）中，expr1 是一个整数表达式。因此传递给 switch 和 case 语句的参数应该是 int、 short、 char 或者 byte。long,string 都不能作用于 swtich。

14、编程题: 用最有效率的方法算出 2 乘以 8 等於几? 【基础】
答： 2 << 3。

15、有没有 length()这个方法? String 有没有 length()这个方法？【基础】
答：数组没有 length()这个方法，有 length 的属性。String 有 length()这个方法。

16、在 JAVA 中，如何跳出当前的多重嵌套循环？【基础】
答：在最外层循环前加 label 标识,然后用 break:label 方法即可跳出多重循环。

17、构造器 Constructor 是否可被 override? 【基础】
答：构造器 Constructor 不能被继承，因此不能重写 Overriding，但可以被重载 Overloading。

18、两个对象值相同(x.equals(y) == true)，但却可有不同的 hash code，这句话对不对? 【基础】
答：不对，有相同的 hash code。

19、是否可以继承 String 类? 【基础】
答：String 类是 final 类，故不可以继承。

20、以下二条语句返回值为 true 的有：
A：“beijing”==“beijing”；
B：“beijing”.equalsIgnoreCase（new String（“beijing”））;【基础】
答：A 和 B 。

21、当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递? 【基础】
答：是值传递。Java 编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。

22、我们在 web 应用开发过程中经常遇到输出某种编码的字符，如 iso8859-1 等，如何输出一个某种编码的字符串？【基础】
答：public String translate(String str){

```
        String tempStr = "";
        try{
            tempStr = new String(str.getBytes("ISO-8859-1"), "GBK");
            tempStr = tempStr.trim();
        }catch (Exception e){
            System.err.println(e.getMessage());
        }
        return tempStr;
    }
```

23、String 和 StringBuffer 的区别? 【基础】
答：JAVA 平台提供了两个类：String 和 StringBuffer，它们可以储存和操作字符串，即包含多个字符的字符数据。这个 String 类提供了数值不可改变的字符串。而这个 StringBuffer 类提供的字符串进行修改。当你知道字符数据要改变的时候你就可以使用 StringBuffer。典型地，你可以使用 StringBuffers 来动态构造字符数据。

24、String, StringBuffer StringBuilder 的区别。【基础】
答：String 的长度是不可变的；
StringBuffer 的长度是可变的，如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用 StringBuffer，如果最后需要 String，那么使用 StringBuffer 的 toString()方法；线程安全；
StringBuilder 是从 JDK 5 开始，为 StringBuffer 该类补充了一个单个线程使用的等价类；通常应该优先使用 StringBuilder 类，因为它支持所有相同的操作，但由于它不执行同步，所以速度更快。

25、Overload 和 Override 的区别。Overloaded 的方法是否可以改变返回值的类型? 【基础】
答：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，我们说该方法被重写 (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。

26、定义类 A 和类 B 如下：【基础】

```
class  A {
    int  a=1;
    double  d=2.0;
    void  show(){
        System.out.println("Class A: a="+a +"\td="+d);
    }
}
class B extends A{
```

```
    float  a=3.0f;
    String  d="Java program.";
    void  show(){
        super.show( );
        System.out.println("Class B:  a="+a +"\td="+d);
    }
}
```

(1) 若在应用程序的 main 方法中有以下语句：

```
A  a=new A();
a.show();
```

则输出的结果如何？

(2) 若在应用程序的 main 方法中定义类 B 的对象 b：

```
A  b=new B();
b.show();
```

则输出的结果如何？

答：输出结果为：

1）Class A: a=1 d=2.0 ；

2）Class A: a=1 d=2.0

Class B: a=3.0 d=Java program。

27、描述一下 JVM 加载 class 文件的原理机制? 【基础】

答：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。

28、char 型变量中能不能存贮一个中文汉字?为什么? 【基础】

答：能够定义成为一个中文的，因为 java 中以 unicode 编码，一个 char 占 16 个字节，所以放一个中文是没问题的。

29、abstract class 和 interface 有什么区别? 【基础】

答：声明方法的存在而不去实现它的类被叫做抽象类（abstract class），它用于要创建一个体现某些基本行为的类，并为该类声明方法，但不能在该类中实现该类的情况。不能创建 abstract 类的实例。然而可以创建一个变量，其类型是一个抽象类，并让它指向具体子类的一个实例。不能有抽象构造函数或抽象静态方法。Abstract 类的子类为它们父类中的所有抽象方法提供实现，否则它们也是抽象类为。取而代之，在子类中实现该方法。知道其行为的其它类可以在类中实现这些方法。接口（interface）是抽象类的变体。新型多继承性可通过实现这样的接口而获得。接口中的所有方法都是抽象的，所有成员变量都是 public static final 的。一个类可以实现多个接口，当类实现特殊接口时，它定义（即将程序体给予）所有这种接口的方法。然后，它可以在实现了该接口的类的任何对象上调用接口的方法。由于有抽象类,它允许使用接口名作为引用变量的类型 。通常的动态联编将生效。引用可以转换到接口类型或从接口类型转换，instanceof 运算符可以用来决定某对象的类是否实现了接口。

30、Static Nested Class 和 Inner Class 的不同？【基础】
答：Static Nested Class 是被声明为静态（static）的内部类，它可以不依赖于外部类实例被实例化。而通常的内部类需要在外部类实例化后才能实例化。

31、java 中会存在内存泄漏吗，请简单描述。【基础】
答：会；存在无用但可达的对象，这些对象不能被 GC 回收，导致耗费内存资源。

32、abstract 的 method 是否可同时是 static,是否可同时是 native，是否可同时是 synchronized? 【基础】
答：都不能。

33、静态变量和实例变量的区别？【基础】
答：静态变量也称为类变量，归全类共有，它不依赖于某个对象，可通过类名直接访问；而实例变量必须依存于某一实例，只能通过对象才能访问到它。

34、是否可以从一个 static 方法内部发出对非 static 方法的调用？【基础】
答：不可以,如果其中包含对象的 method()，不能保证对象初始化。

35、写 clone()方法时，通常都有一行代码，是什么？【基础】
答：Clone 有缺省行为：super.clone()，他负责产生正确大小的空间，并逐位复制。

36、GC 是什么? 为什么要有 GC? 【基础】
答：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。 Java 程序员不用担心内存管理，因为垃圾收集器会自动进行管理。要请求垃圾收集，可以调用下面的方法之一：System.gc() 或 Runtime.getRuntime().gc() 。

37、垃圾回收的优点和原理。并考虑 2 种回收机制。【基础】
答：Java 语言中一个显著的特点就是引入了垃圾回收机制，使 c++程序员最头疼的内存管理的问题迎刃而解，它使得 Java 程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java 中的对象不再有“作用域”的概念,只有对象的引用才有“作用域”。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

38、垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？【基础】
答：对于 GC 来说，当程序员创建对象时，GC 就开始监控这个对象的地址、大小以及使用情况。通常，GC 采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是"可达的"，哪些对象是"不可达的"。当 GC 确定一

些对象为"不可达"时，GC 就有责任回收这些内存空间。可以。程序员可以手动执行 System.gc()，通知 GC 运行，但是 Java 语言规范并不保证 GC 一定会执行。

39、String s=new String(“xyz”);创建了几个 String Object? 【基础】
答：两个对象，一个是"xyx",一个是指向"xyx"的引用对象 s。

40、接口是否可继承接口? 抽象类是否可实现(implements)接口? 抽象类是否可继承实体类(concrete class)? 【基础】
答：接口可以继承接口。抽象类可以实现(implements)接口，抽象类可继承实体类，但前提是实体类必须有明确的构造函数。

41、Java 的接口和 C++的虚类的相同和不同处。【基础】
答：由于 Java 不支持多继承，而有可能某个类或对象要使用分别在几个类或对象里面的方法或属性，现有的单继承机制就不能满足要求。与继承相比，接口有更高的灵活性，因为接口中没有任何实现代码。当一个类实现了接口以后，该类要实现接口里面所有的方法和属性，并且接口里面的属性在默认状态下面都是 public static,所有方法默认情况下是 public.一个类可以实现多个接口。

42、一个“.java”源文件中是否可以包含多个类（不是内部类）？有什么限制？【基础】
答：可以；必须只有一个类名与文件名相同。

43、说出一些常用的类，包，接口，请各举 5 个。【基础】
答：常用的类：BufferedReader BufferedWriter FileReader FileWirter
String Integer;
常用的包：java.lang　java.awt　java.io　java.util　java.sql;
常用的接口：Remote　List　Map　Document　NodeList

44、Anonymous Inner Class (匿名内部类) 是否可以 extends(继承)其它类? 是否可以 implements(实现)interface(接口)? 【基础】
答：可以继承其他类或实现其他接口，在 swing 编程中常用此方式。

45、内部类可以引用他包含类的成员吗？有没有什么限制？【基础】
答：一个内部类对象可以访问创建它的外部类对象的内容。

46、java 中实现多态的机制是什么？【基础】
答：方法的覆盖 Overriding 和重载 Overloading 是 java 多态性的不同表现；覆盖 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。

47、在 java 中一个类被声明为 final 类型，表示了什么意思？【基础】
答：表示该类不能被继承，是顶级类。

48、下面哪些类可以被继承? 【基础】
1）java.lang.Thread (T)

2）java.lang.Number (T)
3）java.lang.Double (F)
4）java.lang.Math (F)
5）java.lang.Void (F)
6）java.lang.Class (F)
7）java.lang.ClassLoader (T)
答：1、2、7 可以被继承。

49、指出下面程序的运行结果：【基础】

```
class A{
    static{
        System.out.print("1");
    }
    public A(){
        System.out.print("2");
    }
}
class B extends A{
    static{
        System.out.print("a");
    }
    public B(){
        System.out.print("b");
    }
}
public class Hello{
    public static void main(String[] ars){
        A ab = new B(); //执行到此处,结果: 1a2b
        ab = new B(); //执行到此处,结果: 1a2b2b
    }
}
```

答：输出结果为 1a2b2b；类的 static 代码段,可以看作是类首次加载(虚拟机加载)执行的代码,而对于类加载,首先要执行其基类的构造,再执行其本身的构造。

50、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?【基础】
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();
    }
}
```

答：输出结果为：

```
    FatherClass Create
    FatherClass Create
    ChildClass Create
```

51、内部类的实现方式? 【基础】

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();
    }
}
```

输出结果为:

```
InterClass Create
OuterClass Create
```

52、关于内部类： 【基础】

```
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();
    }
```

答：答案为 C、E；说明如下：

1）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；

2）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；故 D 错；

3）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 。

53、数据类型之间的转换:

1)如何将数值型字符转换为数字?

2)如何将数字转换为字符?

3)如何取小数点前两位并四舍五入? 【基础】

答：1)调用数值类型相应包装类中的方法 parse***(String)或 valueOf(String)即可返回相应基本类型或包装类型数值;

2)将数字与空字符串相加即可获得其所对应的字符串;另外对于基本类型数字还可调用 String 类中的 valueOf(…)方法返回相应字符串,而对于包装类型数字则可调用其 toString()方法获得相应字符串;

3)可用该数字构造一 java.math.BigDecimal 对象,再利用其 round()方法进行四舍五入到保留小数点后两位,再将其转换为字符串截取最后两位。

54、字符串操作：如何实现字符串的反转及替换？【基础】

答:可用字符串构造一 StringBuffer 对象,然后调用 StringBuffer 中的 reverse 方法即可实现字符串的反转,调用 replace 方法即可实现字符串的替换。

55、编码转换：怎样将 GB2312 编码的字符串转换为 ISO-8859-1 编码的字符串？【基础】

答：示例代码如下:

```
String s1 = "你好";
String s2 = new String(s1.getBytes("GB2312"), "ISO-8859-1");
```

56、写一个函数，要求输入一个字符串和一个字符长度，对该字符串进行分隔。 【基础】

答：函数代码如下：

```
public String[] split(String str, int chars){
    int n = (str.length()+ chars - 1)/chars;
    String ret[] = new String[n];
    for(int i=0; i<n; i++){
      if(i < n-1){
         ret[i] = str.substring(i*chars , (i+1)*chars);
      }else{
         ret[i] = str.substring(i*chars);
      }
    }
    return ret;
}
```

57、写一个函数，2 个参数，1 个字符串，1 个字节数，返回截取的字符串，要求字符串中的中文不能出现乱码：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”。 【基础】

答：代码如下：

```
public String subString(String str, int subBytes) {
    int bytes = 0; // 用来存储字符串的总字节数
    for (int i = 0; i < str.length(); i++) {
        if (bytes == subBytes) {
            return str.substring(0, i);
        }
        char c = str.charAt(i);
        if (c < 256) {
            bytes += 1; // 英文字符的字节数看作 1
        } else {
            bytes += 2; // 中文字符的字节数看作 2
            if(bytes - subBytes == 1){
                return str.substring(0, i);
            }
        }
    }
    return str;
}
```

58、日期和时间：

1)如何取得年月日、小时分秒？

2)如何取得从 1970 年到现在的毫秒数？

3)如何取得某个日期是当月的最后一天？

4)如何格式化日期？ 【基础】

答：1)创建 java.util.Calendar 实例(Calendar.getInstance()),调用其 get()方法传入不同的参数即可获得参数所对应的值,如：

```
calendar.get(Calendar.YEAR);//获得年
```

2)以下方法均可获得该毫秒数:

```
Calendar.getInstance().getTimeInMillis();
System.currentTimeMillis();
```

3)示例代码如下:

```
Calendar time = Calendar.getInstance();
time.set(Calendar.DAY_OF_MONTH,
                time.getActualMaximum(Calendar.DAY_OF_MONTH));
```

4)利用 java.text.DataFormat 类中的 format()方法可将日期格式化。

59、Java 编程,打印昨天的当前时刻。【基础】

答：

```
public class YesterdayCurrent{
    public static void main(String[] args){
        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.DATE, -1);
        System.out.println(cal.getTime());
    }
}
```

60、java 和 javasciprt 的区别。【基础】

答：JavaScript 与 Java 是两个公司开发的不同的两个产品。Java 是 SUN 公司推出的新一代面向对象的程序设计语言，特别适合于 Internet 应用程序开发；而 JavaScript 是 Netscape 公司的产品，其目的是为了扩展 Netscape Navigator 功能,而开发的一种可以嵌入 Web 页面中的基于对象和事件驱动的解释性语言,它的前身是 Live Script；而 Java 的前身是 Oak 语言。下面对两种语言间的异同作如下比较：

1）基于对象和面向对象：

Java 是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象;

JavaScript 是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object Based）和事件驱动（Event Driver）的编程语言。因而它本身提供了非常丰富的内部对象供设计人员使用；

2）解释和编译：

Java 的源代码在执行之前，必须经过编译；

JavaScript 是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行；

3）强类型变量和类型弱变量：

Java 采用强类型变量检查，即所有变量在编译之前必须作声明；

JavaScript 中变量声明，采用其弱类型。即变量在使用前不需作声明，而是解释器在运行时检查其数据类型；

4）代码格式不一样。

61、什么时候用 assert？【中等难度】

答：assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。一般来说，assertion 用于保证程序最基本、关键的正确性。assertion 检查通常在开发和测试时开启。为了提高性能，在软件发布后，

assertion 检查通常是关闭的。在实现中，断言是一个包含布尔表达式的语句，在执行这个语句时假定该表达式为 true；如果表达式计算为 false，那么系统会报告一个 Assertionerror。

断言用于调试目的：

```
assert(a > 0); // throws an Assertionerror if a <= 0
```

断言可以有两种形式：

```
assert Expression1 ;
assert Expression1 : Expression2 ;
```

Expression1 应该总是产生一个布尔值。

Expression2 可以是得出一个值的任意表达式;这个值用于生成显示更多调试信息的 String 消息。

断言在默认情况下是禁用的,要在编译时启用断言,需使用 source 1.4 标记：

```
javac -source 1.4 Test.java
```

要在运行时启用断言，可使用 -enableassertions 或者 -ea 标记。

要在运行时选择禁用断言，可使用 -da 或者 -disableassertions 标记。

要在系统类中启用断言，可使用 -esa 或者 -dsa 标记。还可以在包的基础上启用或者禁用断言。可以在预计正常情况下不会到达的任何位置上放置断言。断言可以用于验证传递给私有方法的参数。不过，断言不应该用于验证传递给公有方法的参数，因为不管是否启用了断言，公有方法都必须检查其参数。不过，既可以在公有方法中，也可以在非公有方法中利用断言测试后置条件。另外，断言不应该以任何方式改变程序的状态。

异常部分：（共 8 题：基础 8 道）

62、Java 中的异常处理机制的简单原理和应用？【基础】

答：当 JAVA 程序违反了 JAVA 的语义规则时，JAVA 虚拟机就会将发生的错误表示为一个异常。违反语义规则包括 2 种情况。一种是 JAVA 类库内置的语义检查。例如数组下标越界,会引发 IndexOutOfBoundsException;访问 null 的对象时会引发 NullPointerException。另一种情况就是 JAVA 允许程序员扩展这种语义检查，程序员可以创建自己的异常，并自由选择在何时用 throw 关键字引发异常。所有的异常都是 java.lang.Thowable 的子类。

63、error 和 exception 有什么区别？【基础】

答：error 表示系统级的错误和程序不必处理的异常，是恢复不是不可能但很困难的情况下的一种严重问题;比如内存溢出,不可能指望程序能处理这样的情况；

exception 表示需要捕捉或者需要程序进行处理的异常，是一种设计或实现问题；也就是说，它表示如果程序运行正常，从不会发生的情况。

64、try {}里有一个 return 语句,那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时候被执行，在 return 前还是后？【基础】

答：会执行，在 return 前执行。

65、JAVA 语言如何进行异常处理，关键字：throws,throw,try,catch,finally 分别代表什么意义？在 try 块中可以抛出异常吗？【基础】

答：Java 通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并

提供了良好的接口。在 Java 中，每个异常都是一个对象，它是 Throwable 类或其它子类的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并进行处理。Java 的异常处理是通过 5 个关键词来实现的：try、catch、throw、throws 和 finally。一般情况下是用 try 来执行一段程序，如果出现异常，系统会抛出（throws）一个异常，这时候你可以通过它的类型来捕捉（catch）它，或最后（finally）由缺省处理器来处理；

try 用来指定一块预防所有“异常”的程序；

catch 子句紧跟在 try 块后面，用来指定你想要捕捉的“异常”的类型；

throw 语句用来明确地抛出一个“异常”；

throws 用来标明一个成员函数可能抛出的各种“异常”；

Finally 为确保一段代码不管发生什么“异常”都被执行一段代码；

可以在一个成员函数调用的外面写一个 try 语句，在这个成员函数内部写另一个 try 语句保护其他代码。每当遇到一个 try 语句，“异常”的框架就放到堆栈上面，直到所有的 try 语句都完成。如果下一级的 try 语句没有对某种“异常"进行处理，堆栈就会展开，直到遇到有处理这种“异常”的 try 语句。

66、运行时异常与一般异常有何异同？【基础】

答：异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java 编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

67、给我一个你最常见到的 runtime exception？【基础】

答：ArithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, ClassCastException, CMMException, ConcurrentModificationException, DOMException, EmptyStackException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException, IllegalStateException, ImagingOpException, IndexOutOfBoundsException, MissingResourceException, NegativeArraySizeException, NoSuchElementException, NullPointerException, ProfileDataException, ProviderException, RasterFormatException, SecurityException, SystemException, UndeclaredThrowableException, UnmodifiableSetException, UnsupportedOperationException

68、final, finally, finalize 的区别？【基础】

答：final：修饰符（关键字）；如果一个类被声明为 final，意味着它不能再派生出新的子类，不能作为父类被继承，因此一个类不能既被声明为 abstract 的，又被声明为 final 的；将变量或方法声明为 final，可以保证它们在使用中不被改变；被声明为 final 的变量必须在声明时给定初值，而在以后的引用中只能读取，不可修改；被声明为 final 的方法也同样只能使用，不能重载。

finally：再异常处理时提供 finally 块来执行任何清除操作；如果抛出一个异常，那么相匹配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。

finalize：方法名；Java 技术允许使用 finalize() 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object 类中定义的，因此所有的类都继承了它。子类覆盖 finalize() 方法以整理系统资源或者执行其他清理工作。finalize() 方法是在垃圾收集器删除对象之前对这个对象调用的。

69、类 Example A 继承 Exception，类 ExampleB 继承 Example A;【基础】
有如下代码片断：

```
try{
    throw new ExampleB(“b”);
}catch (ExampleA  e) {
    System.out.printfln (“ExampleA”);
}catch (Exception  e) {
    System.out.printfln (“Exception”);
}
```

输出的内容应该是：
A：ExampleA　B：Exception　C：b　D：无
答：输出为 A。

集合部分：（共 11 题：基础 11 道）

70、介绍 JAVA 中的 Collection FrameWork(及如何写自己的数据结构)【基础】
答：Collection FrameWork 如下：

```
Collection
 ├List
 │ ├LinkedList
 │ ├ArrayList
 │ └Vector
 │   └Stack
 └Set
Map
 ├Hashtable
 ├HashMap
 └WeakHashMap
```

Collection 是最基本的集合接口，一个 Collection 代表一组 Object，即 Collection 的元素 (Elements)；Map 提供 key 到 value 的映射。

71、List,Set,Map 是否继承自 Collection 接口？【基础】
答：List,Set 是；Map 不是。

72、你所知道的集合类都有哪些？主要方法？【基础】
答：最常用的集合类是 List 和 Map。List 的具体实现包括 ArrayList 和 Vector，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象的元素列表。 List 适用于按数值索引访问元素的情形。 Map 提供了一个更通用的

元素存储方法。 Map 集合类用于存储元素对（称作“键”和“值”），其中每个键映射到一个值。

73、说出 ArrayList,Vector, LinkedList 的存储性能和特性？【基础】
答：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，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 方法（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。

74、Collection 和 Collections 的区别？【基础】
答：Collection 是 java.util 下的接口，它是各种集合的父接口，继承于它的接口主要有 Set 和 List；Collections 是个 java.util 下的类，是针对集合的帮助类，提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

75、HashMap 和 Hashtable 的区别？【基础】
答：二者都实现了 Map 接口，是将惟一键映射到特定的值上；主要区别在于：
1)HashMap 没有排序，允许一个 null 键和多个 null 值,而 Hashtable 不允许;
2)HashMap 把 Hashtable 的 contains 方法去掉了，改成 containsvalue 和 containsKey,因为 contains 方法容易让人引起误解；
3)Hashtable 继承自 Dictionary 类，HashMap 是 Java1.2 引进的 Map 接口的实现；
4)Hashtable 的方法是 Synchronize 的，而 HashMap 不是，在多个线程访问 Hashtable 时，不需要自己为它的方法实现同步，而 HashMap 就必须为之提供外同步。
Hashtable 和 HashMap 采用的 hash/rehash 算法大致一样，所以性能不会有很大的差异。

76、Arraylist 与 Vector 区别？【基础】
答：就 ArrayList 与 Vector 主要从二方面来说：
1）同步性：Vector 是线程安全的（同步），而 ArrayList 是线程序不安全的；
2）数据增长：当需要增长时,Vector 默认增长一倍，而 ArrayList 却是一半。

77、List、Map、Set 三个接口，存取元素时，各有什么特点？【基础】
答：List 以特定次序来持有元素，可有重复元素。Set 无法拥有重复元素,内部排序。Map 保存 key-value 值，value 可多值。

78、Set 里的元素是不能重复的，那么用什么方法来区分重复与否呢？是用==还是 equals()？它们有何区别？【基础】
答：Set 里的元素是不能重复的，用 equals ()方法来区分重复与否。覆盖 equals()方法用来判断对象的内容是否相同，而”==”判断地址是否相等,用来决定引用值是否指向同一对象。

79、用程序给出随便大小的 10 个数，序号为 1-10，按从小到大顺序输出，并输出相应的序号。【基础】

答：代码如下：

```
package test;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import java.util.Random;
public class RandomSort {
    public static void printRandomBySort() {
        Random random = new Random(); // 创建随机数生成器
        List list = new ArrayList();
        // 生成 10 个随机数，并放在集合 list 中
        for (int i = 0; i < 10; i++) {
            list.add(random.nextInt(1000));
        }
        Collections.sort(list); // 对集合中的元素进行排序
        Iterator it = list.iterator();
        int count = 0;
        while (it.hasNext()) { // 顺序输出排序后集合中的元素
            System.out.println(++count + ":  " + it.next());
        }
    }
    public static void main(String[] args) {
        printRandomBySort();
    }
}
```

80、用 JAVA 实现一种排序，JAVA 类实现序列化的方法？ 在 COLLECTION 框架中，实现比较要实现什么样的接口？【基础】

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(){
        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();
    }
}
```

JAVA 类实现序例化的方法是实现 java.io.Serializable 接口；

Collection 框架中实现比较要实现 Comparable 接口和 Comparator 接口。

线程部分：（共 10 题：基础 7 道，中等难度 3 道）

81、sleep() 和 wait() 有什么区别? 【基础】

答：sleep 是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用 sleep 不会释放对象锁。wait 是 Object 类的方法，对此对象调用 wait 方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出 notify 方法（或 notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态。

82、当一个线程进入一个对象的一个 synchronized 方法后，其它线程是否可进入此对象的其它方法? 【基础】
答：其它线程只能访问该对象的其它非同步方法，同步方法则不能进入。

83、请说出你所知道的线程同步的方法。【基础】
答：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 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；
notityAll():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。

84、多线程有几种实现方法,都是什么?同步有几种实现方法,都是什么? 【基础】
答：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。

85、同步和异步有何异同，在什么情况下分别使用他们？举例说明。【基础】
答：如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

86、启动一个线程是用 run()还是 start()?【基础】
答：启动一个线程是调用 start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由 JVM 调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。

87、 线程的基本概念、线程的基本状态以及状态之间的关系？【基础】
答：线程指在程序执行过程中，能够执行程序代码的一个执行单位，每个程序至少都有一个线程，也就是程序本身；
Java 中的线程有四种状态分别是：运行、就绪、挂起、结束。

88、简述 synchronized 和 java.util.concurrent.locks.Lock 的异同 ？【中等难度】
答：主要相同点：Lock 能完成 synchronized 所实现的所有功能；
主要不同点：Lock 有比 synchronized 更精确的线程语义和更好的性能。synchronized 会自动释放锁，而 Lock 一定要求程序员手工释放，并且必须在 finally 从句中释放。

89、java 中有几种方法可以实现一个线程？用什么关键字修饰同步方法? stop()和 suspend()方法为何不推荐使用？【中等难度】
答：有两种实现方法，分别是继承 Thread 类与实现 Runnable 接口；

用 synchronized 关键字修饰同步方法；

反对使用 stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在；

suspend()方法容易发生死锁。调用 suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被“挂起”的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。故不应该使用 suspend()，而应在自己的 Thread 类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用 wait()命其进入等待状态。若标志指出线程应当恢复，则用一个 notify()重新启动线程。

90、设计 4 个线程，其中两个线程每次对 j 增加 1，另两个线程对 j 每次减少 1。写出程序。【中等难度】

答：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 TestThread {
   private int j;
   public TestThread(int j) {this.j = j;}
   private synchronized void inc(){
       j++;
       System.out.println(j + "--Inc--" +
                          Thread.currentThread().getName());
   }
   private synchronized void dec(){
       j--;
       System.out.println(j + "--Dec--" +
                          Thread.currentThread().getName());
   }
   public void run() {
       (new Dec()).start();
       new Thread(new Inc()).start();
       (new Dec()).start();
       new Thread(new Inc()).start();
   }
   class Dec extends Thread {
       public void run() {
           for(int i=0; i<100; i++){
               dec();
           }
       }
   }
   class Inc implements Runnable {
       public void run() {
           for(int i=0; i<100; i++){
```

```
                    inc();
                }
            }
        }
        public static void main(String[] args) {
            (new TestThread(5)).run();
        }
}
```

IO 流及 Socket 部分：（共 5 题：基础 5 道）

91、什么是 java 序列化，如何实现 java 序列化？【基础】
答：序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题；
序列化的实现：将需要被序列化的类实现 Serializable 接口，该接口没有需实现的方法，implements Serializable 只是为了标注该对象是可被序列化的，然后使用一个输出流(如 FileOutputStream)来构造一个 ObjectOutputStream(对象流)对象，接着，使用 ObjectOutputStream 对象的 writeObject(Object obj)方法就可以将参数为 obj 的对象写出(即保存其状态)，要恢复的话则用输入流。

92、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？【基础】
答：字节流，字符流。字节流继承于 InputStream、OutputStream，字符流继承于 Reader、Writer。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。

93、文件和目录（IO）操作：
1)如何列出某个目录下的所有文件？
2)如何列出某个目录下的所有子目录？
3)如何判断一个文件或目录是否存在？
4)如何读写文件？【基础】
答：1)示例代码如下:

```
File file = new File("e:\\总结");
File[] files = file.listFiles();
for(int i=0; i<files.length; i++){
    if(files[i].isFile()) System.out.println(files[i]);
}
```

2)示例代码如下:

```
File file = new File("e:\\总结");
File[] files = file.listFiles();
for(int i=0; i<files.length; i++){
    if(files[i].isDirectory()) System.out.println(files[i]);
}
```

3)创建 File 对象,调用其 exsit()方法即可返回是否存在,如:

```
System.out.println(new File("d:\\t.txt").exists());
```

4)示例代码如下:

```
//读文件:
FileInputStream fin = new FileInputStream("e:\\tt.txt");
byte[] bs = new byte[100];
while(true){
    int len = fin.read(bs);
    if(len <= 0) break;
    System.out.print(new String(bs,0,len));
}
fin.close();
//写文件:
FileWriter fw = new FileWriter("e:\\test.txt");
fw.write("hello world!" + System.getProperty("line.separator"));

fw.write("你好！北京！");
fw.close();
```

94、写一个方法,输入一个文件名和一个字符串,统计这个字符串在这个文件中出现的次数。【基础】

答：代码如下：

```
public int countWords(String file, String find) throws Exception
{
    int count = 0;
    Reader in = new FileReader(file);
    int c;
    while ((c = in.read()) != -1) {
        while (c == find.charAt(0)) {
            for (int i = 1; i < find.length(); i++) {
                c = in.read();
                if (c != find.charAt(i)) break;
                if (i == find.length() - 1) count++;
            }
        }
    }
    return count;
}
```

95、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？【基础】

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();
        }
    }
```

二、OOA/D 与 UML 部分：（共 6 题：基础 2 道，中等难度 4 道）

96、UML 是什么？常用的几种图？【基础】
答：UML 是标准建模语言；常用图包括：用例图,静态图(包括类图、对象图和包图),行为图,交互图(顺序图,合作图),实现图。

97、编程题: 写一个 Singleton 出来。【基础】
答：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。举例：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。
第一种形式:

```
public class Singleton {
    private Singleton(){}
    private static Singleton instance = new Singleton();
    public static Singleton getInstance(){
        return instance;
    }
```

```
}
```
第二种形式:
```
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance(){
        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 。

98、说说你所熟悉或听说过的 j2ee 中的几种常用模式?及对设计模式的一些看法。【中等难度】

答：Session Facade Pattern：使用 SessionBean 访问 EntityBean；

Message Facade Pattern：实现异步调用；

EJB Command Pattern：使用 Command JavaBeans 取代 SessionBean，实现轻量级访问；

Data Transfer Object Factory：通过 DTO Factory 简化 EntityBean 数据提供特性；

Generic Attribute Access：通过 AttibuteAccess 接口简化 EntityBean 数据提供特性；

Business Interface：通过远程（本地）接口和 Bean 类实现相同接口规范业务逻辑一致性；

EJB 架构的设计好坏将直接影响系统的性能、可扩展性、可维护性、组件可重用性及开发效率。项目越复杂,项目队伍越庞大则越能体现良好设计的重要性 。

99、Java 中常用的设计模式？说明工厂模式？【中等难度】

答:Java 中的 23 种设计模式:Factory(工厂模式),Builder(建造模式), Factory Method(工厂方法模式),Prototype(原始模型模式),Singleton(单例模式), Facade(门面模式),Adapter(适配器模式), Bridge(桥梁模式), Composite(合成模式),Decorator(装饰模式), Flyweight(享元模式), Proxy(代理模式),Command(命令模式), Interpreter(解释器模式), Visitor(访问者模式),Iterator(迭代子模式), Mediator(调停者模式), Memento(备忘录模式),Observer(观察者模式),State(状态模式),Strategy(策略模式),Template Method(模板方法模式), Chain Of Responsibleity(责任链模式)。

工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某一个类的实例,通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实

例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

100、开发中都用到了那些设计模式?用在什么场合? 【中等难度】
答：每个模式都描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心。通过这种方式，你可以无数次地使用那些已有的解决方案，无需在重复相同的工作。主要用到了 MVC 的设计模式，用来开发 JSP/Servlet 或者 J2EE 的相关应用；及简单工厂模式等。

101、你对软件开发中迭代的含义的理解;【中等难度】
答：软件开发中，各个开发阶段不是顺序执行的，应该是并行执行,也就是迭代的意思。这样对于开发中的需求变化，及人员变动都能得到更好的适应。

三、XML 部分：（共 4 题：基础 1 道，中等难度 1 道，较难 2 道）

102、XML 文档定义有几种形式？它们之间有何本质区别？解析 XML 文档有哪几种方式？【基础】
答：1）两种形式：dtd 以及 schema；
2）本质区别：schema 本身是 xml 的，可以被 XML 解析器解析(这也是从 DTD 上发展 schema 的根本目的)；
3）解析方式：有 DOM,SAX,STAX 等：
DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存,适合对 XML 的随机访问；
SAX:不同于 DOM,SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问；
STAX:Streaming API for XML (StAX)。

103、你在项目中用到了 xml 技术的哪些方面?如何实现的? 【中等难度】
答:用到了数据存贮，信息配置两方面。在做数据交换平台时，将不能数据源的数据组装成 XML 文件，然后将 XML 文件压缩打包加密后通过网络传送给接收者，接收解密与解压缩后再同 XML 文件中还原相关信息进行处理。在做软件配置时，利用 XML 可以很方便的进行，软件的各种配置参数都存贮在 XML 文件中。

104、用 jdom 解析 xml 文件时如何解决中文问题?如何解析? 【较难】
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());
            }
        }
    }
```

105、编程用 JAVA 解析 XML 的方式。【较难】

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);
    }
}
```

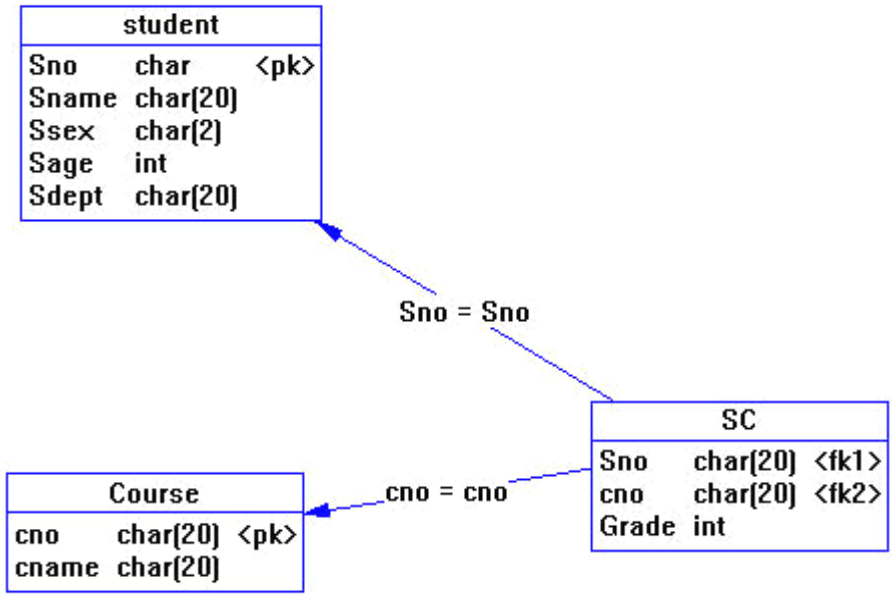
JSP 内容显示源码,SaxXml.jsp:

```
<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>
```

四、数据库及 SQL 部分：（共 4 题：基础 3 道，中等难度 1 道）

106、有 3 个表（15 分钟）：【基础】
Student 学生表（学号，姓名，性别，年龄，组织部门）
Course 课程表（编号，课程名称）
Sc 选课表（学号，课程编号，成绩）
表结构如下：



1）写一个 SQL 语句，查询选修了’计算机原理’的学生学号和姓名（3 分钟）
2）写一个 SQL 语句，查询’周星驰’同学选修了的课程名字（3 分钟）
3）写一个 SQL 语句，查询选修了 5 门课程的学生学号和姓名（9 分钟）
答：1）SQL 语句如下：

```
select stu.sno, stu.sname from Student stu
where (select count(*) from sc where sno=stu.sno and cno =
        (select cno from Course where cname='计算机原理')) != 0;
```

2）SQL 语句如下：

```
select cname from Course
where cno in ( select cno from sc where sno =
              (select sno from Student where sname='周星驰'));
```

3）SQL 语句如下：

```
select stu.sno, stu.sname from student stu
where (select count(*) from sc where sno=stu.sno) = 5;
```

107、有三张表,学生表 S,课程 C,学生课程表 SC,学生可以选修多门课程,一门课程可以被多个学生选修,通过 SC 表关联。【基础】
1）写出建表语句；
2）写出 SQL 语句,查询选修了所有选修课程的学生；
3）写出 SQL 语句,查询选修了至少 5 门以上的课程的学生。
答：1）建表语句如下（mysql 数据库）：

```
create table s(id integer primary key, name varchar(20));
create table c(id integer primary key, name varchar(20));
create table sc(
   sid integer references s(id),
   cid integer references c(id),
   primary key(sid,cid)
);
```

2）SQL 语句如下：

```
select stu.id, stu.name from s stu
    where (select count(*) from sc where sid=stu.id)
        = (select count(*) from c);
```

3）SQL 语句如下：

```
select stu.id, stu.name from s stu
    where (select count(*) from sc where sid=stu.id)>=5;
```

108、数据库表(Test)结构如下：【基础】

ID	NAME	AGE	MANAGER(所属主管人 ID)
106	A	30	104
109	B	19	104
104	C	20	111
107	D	35	109
112	E	25	120
119	F	45	NULL

要求:列出所有年龄比所属主管年龄大的人的 ID 和名字?
答：SQL 语句如下：

```
select employee.name from test employee
where employee.age > (select manager.age from test manager
                      where manager.id=employee.manager);
```

109、有如下两张表:【中等难度】
表 city:　　表 state:

CityNo	CityName	StateNo
BJ	北京	（Null）
SH	上海	（Null）
GZ	广州	GD
DL	大连	LN

State No	State Name
GD	广东
LN	辽宁
SD	山东
NMG	内蒙古

欲得到如下结果：

City Name	State No	State Name
BJ	北京	（Null）
DL	大连	LN 辽宁
GZ	广州	GD 广东
SH	上海	（Null）

写相应的 SQL 语句。

答：SQL 语句为：

```
SELECT C.CITYNO, C.CITYNAME, C.STATENO, S.STATENAME
FROM CITY C, STATE S
WHERE C.STATENO=S.STATENO(+)
ORDER BY(C.CITYNO);
```

五、JDBC 及 Hibernate:（共 12 题：基础 10 道，中等难度 2 道）

110、数据库，比如 100 用户同时来访，要采取什么技术解决？【基础】

答：可采用连接池。

111、什么是 ORM？【基础】

答：对象关系映射（Object—Relational Mapping，简称 ORM）是一种为了解决面向对象与面向关系数据库存在的互不匹配的现象的技术；简单的说，ORM 是通过使用描述对象和数据库之间映射的元数据，将 java 程序中的对象自动持久化到关系数据库中；本质上就是将数据从一种形式转换到另外一种形式。

112、Hibernate 有哪 5 个核心接口？【基础】

答：Configuration 接口：配置 Hibernate，根据其启动 hibernate，创建 SessionFactory 对象；

SessionFactory 接口：初始化 Hibernate，充当数据存储源的代理，创建 session 对象，sessionFactory 是线程安全的，意味着它的同一个实例可以被应用的多个线程共享，是重量级、二级缓存；

Session 接口：负责保存、更新、删除、加载和查询对象，是线程不安全的，避免多个线程共享同一个 session，是轻量级、一级缓存；

Transaction 接口：管理事务；

Query 和 Criteria 接口：执行数据库的查询。

113、关于 hibernate: 【基础】

1）在 hibernate 中，在配置文件呈标题一对多，多对多的标签是什么；

2）Hibernate 的二级缓存是什么；

3）Hibernate 是如何处理事务的；

答：1）一对多的标签为<one-to-many> ；多对多的标签为<many-to-many>；

2）sessionFactory 的缓存为 hibernate 的二级缓存；

3）Hibernate 的事务实际上是底层的 JDBC Transaction 的封装或者是 JTA Transaction 的封装；默认情况下使用 JDBCTransaction。

114、Hibernate 的应用（Hibernate 的结构）？【基础】
答：//首先获得 SessionFactory 的对象
```
SessionFactory sessionFactory = new Configuration().configure().
                                        buildSessionFactory();
//然后获得 session 的对象
Session session = sessionFactory.openSession();
//其次获得 Transaction 的对象
Transaction tx = session.beginTransaction();
//执行相关的数据库操作:增,删,改,查
session.save(user); //增加, user 是 User 类的对象
session.delete(user); //删除
session.update(user); //更新
Query query = session.createQuery(“from User”); //查询
List list = query.list();
//提交事务
tx.commit();
//如果有异常,我们还要作事务的回滚,恢复到操作之前
tx.rollback();
//最后还要关闭 session,释放资源
session.close();
```

115、什么是重量级？什么是轻量级？【基础】
答：轻量级是指它的创建和销毁不需要消耗太多的资源，意味着可以在程序中经常创建和销毁 session 的对象；重量级意味不能随意的创建和销毁它的实例，会占用很多的资源。

116、数据库的连接字符串？【基础】
答：MS SQL Server
```
//第二种连接方式
Class.forName(“com.microsoft.jdbc.sqlserver.SQLServerDriver”).
          newInstance();
conn = DriverManager.getConnection(“jdbc:Microsoft:sqlserver
              ://localhost:1433;DatabaseName=pubs”,”sa”,””);
//Oracle
Class.forName(“oracle.jdbc.driver.OracleDriver”).newInstance();
conn = DriverManager.getConnection(“jdbc:oracle:thin:
                @localhost:1521:sid”, uid, pwd);
//Mysql
Class.forName(“org.git.mm.mysql.Driver”).newInstance();
conn = DriverManager.getConnection(“jdbc:mysql
              ://localhost:3306/pubs”,”root”,””);
处理中文的问题:
jdbc:mysql://localhost:3306/pubs?useUnicode=true
                                  &characterEncoding=GB2312
```

117、事务处理？【基础】
答：Connection 类中提供了 3 个事务处理方法：
setAutoCommit(Boolean autoCommit):设置是否自动提交事务，默认为自动提交事务，即为 true，通过设置 false 禁止自动提交事务；
commit():提交事务；
rollback():回滚事务。

118、Java 中访问数据库的步骤？Statement 和 PreparedStatement 之间的区别？【基础】
答：Java 中访问数据库的步骤如下：
1）注册驱动；
2）建立连接；
3）创建 Statement；
4）执行 sql 语句；
5）处理结果集（若 sql 语句为查询语句）；
6）关闭连接。
PreparedStatement 被创建时即指定了 SQL 语句，通常用于执行多次结构相同的 SQL 语句。

119、用你熟悉的语言写一个连接 ORACLE 数据库的程序,能够完成修改和查询工作。【基础】
答：JDBC 示例程序如下：

```
public void testJdbc(){
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    try{
        //step1：注册驱动；
        Class.forName("oracle.jdbc.driver.OracleDriver");
        //step 2：获取数据库连接；
        con=DriverManager.getConnection(
              "jdbc:oracle:thin:@192.168.0.39:1521:TARENADB",
              "sd0605","sd0605");
        /************************查 询************************/
        //step 3：创建 Statement；
        String sql = "SELECT id, fname, lname, age, FROM
Person_Tbl";
        ps = con.prepareStatement(sql);
        //step 4 ：执行查询语句，获取结果集；
        rs = ps.executeQuery();
        //step 5：处理结果集—输出结果集中保存的查询结果；
        while (rs.next()){
            System.out.print("id = " + rs.getLong("id"));
            System.out.print(" , fname = " +
```

```
rs.getString("fname"));
                System.out.print(" , lname = " +
rs.getString("lname"));
                System.out.print(" , age = " + rs.getInt("age"));
            }
            /************************JDBC 修 改********************/
            sql = "UPDATE Person_Tbl SET age=23 WHERE id = ?";
            ps = con.prepareStatement(sql);
            ps.setLong(1, 88);
            int rows = ps.executeUpdate();
            System.out.println(rows + " rows affected.");
        } catch (Exception e){
           e.printStackTrace();
        } finally{
            try{
                con.close(); //关闭数据库连接，以释放资源。
            } catch (Exception e1) {
            }
        }
    }
```

120、JDBC，Hibernate 分页怎样实现？【中等难度】
答：方法分别为：
1) Hibernate 的分页：

```
Query query = session.createQuery("from Student");
query.setFirstResult(firstResult);//设置每页开始的记录号
query.setMaxResults(resultNumber);//设置每页显示的记录数
Collection students = query.list();
```

2) JDBC 的分页：根据不同的数据库采用不同的 sql 分页语句
例如: Oracle 中的 sql 语句为: "SELECT * FROM (SELECT a.*, rownum r FROM TB_STUDENT) WHERE r between 2 and 10" 查询从记录号 2 到记录号 10 之间的所有记录

121、在 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。 【中等难度】
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%>页
<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>
```

六、Web 部分：（共题：基础 40 道，基础 37 道，中等难度 3 道）

122、说出 Servlet 的生命周期，并说出 Servlet 和 CGI 的区别？【基础】
答：Web 容器加载 Servlet 并将其实例化后，Servlet 生命周期开始，容器运行其 init 方法进行 Servlet 的初始化，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。与 cgi 的区别在于 servlet 处于服务器进程中，它通过多线程方式运行其 service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁，而 CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于 servlet。

123、Servlet 的基本架构。【基础】
答：
```
public class ServletName extends HttpServlet {
        public void doPost(HttpServletRequest request,
                           HttpServletResponse response)
                           throws ServletException, IOException {
        }
        public void doGet(HttpServletRequest request,
                           HttpServletResponse response)
                           throws ServletException, IOException {
        }
    }
```

124、forward 和 redirect 的区别？【基础】
答：forward 是容器中控制权的转向，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，并且从浏览器的地址栏中可以看到跳转后的链接地址。前者更加高效，在前者可以满足需要时，尽量使用 forward()方法，并且，这样也有助于隐藏实际的链接；在有些情况下，比如，需要跳转到一个其它服务器上的资源，则必须使用 sendRedirect()方法。

125、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别？【基础】
答: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,它不会检查所含文件的变化,适用于包含静态页面<%@ include file="included.htm" %>

126、说出数据连接池的工作机制是什么？【基础】
答：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。

127、JSP 的内置对象及方法？【基础】
答：request 表示 HttpServletRequest 对象。它包含了有关浏览器请求的信息,并且提供了几个用于获取 cookie, header 和 session 数据的有用的方法。

response 表示 HttpServletResponse 对象，并提供了几个用于设置送回 浏览器的响应的方法（如 cookies,头信息等）。

out 对象是 javax.jsp.JspWriter 的一个实例，并提供了几个方法使你能用于向浏览器回送输出结果。

pageContext 表示一个 javax.servlet.jsp.PageContext 对象。它是用于方便存取各种范围的名字空间、servlet 相关的对象的 API，并且包装了通用的

servlet 相关功能的方法。

session 表示一个请求的 javax.servlet.http.HttpSession 对象。Session 可以存贮用户的状态信息。

application 表示一个 javax.servle.ServletContext 对象。这有助于查找有关 servlet 引擎和 servlet 环境的信息。

config 表示一个 javax.servlet.ServletConfig 对象。该对象用于存取 servlet 实例的初始化参数。

page 表示从该页面产生的一个 servlet 实例。

128、JSP 的常用指令？【基础】

答：<%@page language=”java” contenType=”text/html;charset=gb2312” session=”true” buffer=”64kb” autoFlush=”true” isThreadSafe=”true” info=”text” errorPage=”error.jsp” isErrorPage=”true” isELIgnored=”true” pageEncoding=”gb2312” import=”java.sql.*”%>

isErrorPage：是否能使用 Exception 对象；isELIgnored：是否忽略 EL 表达式；

<%@include file=”filename”%>

<%@taglib prefix=”c”uri=”http://……”%>

129、jsp 有哪些动作?作用分别是什么？【基础】

答：JSP 共有以下 6 种基本动作：

jsp:include：在页面被请求的时候引入一个文件；

jsp:useBean：寻找或者实例化一个 JavaBean。；

jsp:setProperty：设置 JavaBean 的属性。；

jsp:getProperty：输出某个 JavaBean 的属性；

jsp:forward：把请求转到一个新的页面；

jsp:plugin：根据浏览器类型为 Java 插件生成 OBJECT 或 EMBED 标记。

130、jsp 有哪些内置对象?作用分别是什么？【基础】

答：JSP 共有以下 9 种基本内置组件（可与 ASP 的 6 种内部组件相对应）：

request：用户端请求，此请求会包含来自 GET/POST 请求的参数；

response：网页传回用户端的回应；

pageContext：网页的属性是在这里管理；

session：与请求有关的会话期；

application：servlet 正在执行的内容；

out：用来传送回应的输出；

config：servlet 的构架部件；

page：JSP 网页本身；

exception：针对错误网页，未捕捉的例外。

131、get 和 post 的区别？【基础】

答：Form 中的 get 和 post 方法，在数据传输过程中分别对应了 HTTP 协议中的 GET 和 POST 方法。二者主要区别如下：

1）Get 是用来从服务器上获得数据，而 Post 是用来向服务器上传递数据；

2）Get 将表单中数据按照 variable=value 的形式，添加到 action 所指向的 URL 后面，并且两者使用“?”连接，而各个变量之间使用“&”连接；Post 是将

表单中的数据放在 form 的数据体中，按照变量和值相对应的方式，传递到 action 所指向 URL；

3）Get 是不安全的，因为在传输过程，数据被放在请求的 URL 中；Post 的所有操作对用户来说都是不可见的；

4）Get 传输的数据量小，这主要是因为受 URL 长度限制；而 Post 可以传输大量的数据，所以在上传文件只能使用 Post；

5)Get 限制 Form 表单的数据集必须为 ASCII 字符，而 Post 支持整个 ISO10646 字符集；

6）Get 是 Form 的默认方法。

132、什么情况下调用 doGet()和 doPost()？【基础】

答：Jsp 页面中的 form 标签里的 method 属性为 get 时调用 doGet()，为 post 时调用 doPost()。

133、如何从 form 表单中得取 checkbox 的值；【基础】

答：可在页面把 checkbox 的 name 属性取同一个，value 属性取每个条目的 id，后台用 getParamter(“name”)能取到 checkbox 的一组值。

134、页面中有一个命名为 bank No 的下拉列表，写脚本获取当前选项的索引值。【基础】

答：用 java 或 javaScript 的处理方式分别如下：

Java： request.getParameter(“bank No”);

javaScript：

```
var selectItems = document.getElementsByName(“bank No”);
selectItems[0].value;
```

135、javascript 常用的方面；【基础】

答：常用于数据输入校验和页面特殊效果等。

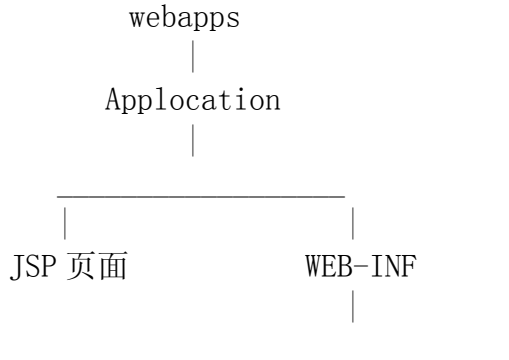
136、常用的 web 容器和开发工具；【基础】

答：最常用的容器包括：tomcat、weblogic；

开发工具有：eclipse,jbuilder。

137、请画出 Servlet 2.2 以上 Web Application 的基本目录结构（2 分钟）【基础】

答：目录结构如下图所示：



```
        |          |          |
    classes      lib     web.xml
```

138、JSP 和 Servlet 有哪些相同点和不同点，他们之间的联系是什么？【基础】
答：JSP 是 Servlet 技术的扩展，本质上是 Servlet 的简易方式，更强调应用的外表表达。JSP 编译后是"类 servlet"。Servlet 和 JSP 最主要的不同点在于，Servlet 的应用逻辑是在 Java 文件中,并且完全从表示层中的 HTML 里分离开来。而 JSP 的情况是 Java 和 HTML 可以组合成一个扩展名为.jsp 的文件。JSP 侧重于视图，Servlet 主要用于控制逻辑。

139、jsp 的四种范围？【基础】
答：a.page 是代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java servlet 类（可以带有任何的 include 指令，但是没有 include 动作）表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面

b.request 是代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面,涉及多个 Web 组件(由于 forward 指令和 include 动作的关系)

c.session 是代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常会跨越多个客户机请求

d.application 是代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域。

140、Request 对象的主要方法? 【基础】
答：setAttribute(String name,Object)：设置名字为 name 的属性值
getAttribute(String name)：返回由 name 指定的属性值
getAttributeNames()：返回 request 对象所有属性的名字集合(枚举)
getCookies()：返回客户端的所有 Cookie 对象，结果是一个 Cookie 数组
getCharacterEncoding()：返回请求中的字符编码方式
getContentLength()：返回请求的 Body 的长度
getHeader(String name)：获得 HTTP 协议定义的文件头信息
getHeaders(String name)：返回指定名的 request Header 的所有值(枚举)
getHeaderNames()：返回所有 request Header 的名字(枚举)
getInputStream()：返回请求的输入流，用于获得请求中的数据
getMethod()：获得客户端向服务器端传送数据的方法
getParameter(String name)：获得客户端请求中传送的 name 指定的参数值
getParameterNames()：获得客户端传送给服务器端的所有参数的名字(枚举)
getParameterValues(String name)：获得有 name 指定的参数的所有值
getProtocol()：获取客户端向服务器端传送数据所依据的协议名称
getQueryString()：获得查询字符串
getRequestURI()：获取发出请求字符串的客户端地址
getRemoteAddr()：获取客户端的 IP 地址
getRemoteHost()：获取客户端的名字
getSession([Boolean create])：返回和请求相关 Session
getServerName()：获取服务器的名字

getServletPath()：获取客户端所请求的脚本文件的路径
getServerPort()：获取服务器的端口号
removeAttribute(String name)：删除请求中的一个属性

141、如何实现 servlet 的单线程模式？【基础】
答：<%@page isThreadSafe=”false”%>

142、页面间对象传递的方法。【基础】
答：request,session,application,cookie 等。

143、详细描述 MVC。【基础】
答：基于 Java 的 Web 应用系统采用 MVC 架构模式，即 model（模型）、view（视图）、control（控制）分离设计；这是目前 WEB 应用服务系统的主流设计方向。
Model：即处理业务逻辑的模块，每一种处理一个模块；
View：负责页面显示，显示 MODEL 处理结果给用户，主要实现数据到页面转换过程；
Control：负责每个请求的分发，把 FORM 数据传递给 MODEL 处理，把处理结果的数据传递给 VIEW 显示。

144、MVC 的各个部分都有那些技术来实现?如何实现？【基础】
答：MVC 是 Model－View－Controller 的简写。"Model" 代表的是应用的业务逻辑（通过 JavaBean，EJB 组件实现）， "View" 是应用的表示面（由 JSP 页面产生），"Controller" 是提供应用的处理过程控制（一般是一个 Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用。

145、应用服务器有那些？【基础】
答：BEA WebLogic Server, IBM WebSphere Application Server, Oracle9i Application Server, JBoss, Tomcat。

146、Servlet 执行时一般实现哪几个方法？【基础】
答：public void init(ServletConfig config)
public ServletConfig getServletConfig()
public String getServletInfo()
public void service(ServletRequest request,ServletResponse response)
public void destroy()

147、struts 的入口类？【基础】
答：是 ActionServlet，所有的 struts 请求都经由该类转发处理。

148、STRUTS 的应用(如 STRUTS 架构)？【基础】
答：Struts 是采用 Java Servlet/JavaServer Pages 技术开发 Web 应用程序的开放源码的 framework。 采用 Struts 能开发出基于 MVC(Model-View-Controller)设计模式的应用构架。 Struts 有如下的主要功能：

1)包含一个 controller servlet,能将用户的请求发送到相应的 Action 对象；
2)JSP 自由 tag 库，并且在 controller servlet 中提供关联支持，帮助开发人员创建交互式表单应用；
3)提供了一系列实用对象：XML 处理、通过 Java reflection APIs 自动处理 JavaBeans 属性、国际化的提示和消息。

149、几种会话跟踪技术？【基础】
答：cookie、URL 重写、设置表单隐藏域。

150、BS 与 CS 的联系与区别？【基础】
答：C/S 是 Client/Server 的缩写，是客户机与服务器结构的应用程序，服务器通常采用高性能的 PC、工作站或小型机，并采用大型数据库系统，如 Oracle、Sybase、Informix 或 SQL Server。客户端需要安装专用的客户端软件。B/S 是 Brower/Server 的缩写，是浏览器和服务器结构的应用程序，即 Web 应用程序，客户机上只要安装一个浏览器（Browser），如 Netscape Navigator 或 Internet Explorer，服务器安装 Oracle、Sybase、Informix 或 SQL Server 等数据库。在这种结构下,用户界面完全通过 WWW 浏览器实现,一部分事务逻辑在前端实现，但是主要事务逻辑在服务器端实现。浏览器通过 Web Server 同数据库进行数据交互。
C/S 与 B/S 区别：
1)硬件环境不同:
C/S 一般建立在专用的网络上, 小范围里的网络环境, 局域网之间再通过专门服务器提供连接和数据交换服务；
B/S 建立在广域网之上的, 不必是专门的网络硬件环境,例与电话上网, 租用设备. 信息自己管理. 有比 C/S 更强的适应范围, 一般只要有操作系统和浏览器就行 ；
2)对安全要求不同：
C/S 一般面向相对固定的用户群, 对信息安全的控制能力很强. 一般高度机密的信息系统采用 C/S 结构适宜. 可以通过 B/S 发布部分可公开信息；
B/S 建立在广域网之上, 对安全的控制能力相对弱, 可能面向不可知的用户；
3)对程序架构不同：
C/S 程序可以更加注重流程, 可以对权限多层次校验, 对系统运行速度可以较少考虑；
B/S 对安全以及访问速度的多重的考虑, 建立在需要更加优化的基础之上. 比 C/S 有更高的要求 B/S 结构的程序架构是发展的趋势, 从 MS 的.Net 系列的 BizTalk 2000 Exchange 2000 等, 全面支持网络的构件搭建的系统. SUN 和 IBM 推的 JavaBean 构件技术等,使 B/S 更加成熟；
4)软件重用不同：
C/S 程序可以不可避免的整体性考虑, 构件的重用性不如在 B/S 要求下的构件的重用性好；
B/S 对的多重结构,要求构件相对独立的功能. 能够相对较好的重用.就入买来的餐桌可以再利用,而不是做在墙上的石头桌子；
5)系统维护不同：

C/S 程序由于整体性，必须整体考察，处理出现的问题以及系统升级．升级难．可能是再做一个全新的系统；

B/S 构件组成,方面构件个别的更换,实现系统的无缝升级．系统维护开销减到最小.用户从网上自己下载安装就可以实现升级；

6)处理问题不同：

C/S 程序可以处理用户面固定，并且在相同区域，安全要求高需求，与操作系统相关．应该都是相同的系统；

B/S 建立在广域网上，面向不同的用户群，分散地域，这是 C/S 无法作到的．与操作系统平台关系最小；

7)用户接口不同：

C/S 多是建立的 Window 平台上,表现方法有限,对程序员普遍要求较高；

B/S 建立在浏览器上，有更加丰富和生动的表现方式与用户交流．并且大部分难度减低,减低开发成本；

8)信息流不同：

C/S 程序一般是典型的中央集权的机械式处理，交互性相对低；

B/S 信息流向可变化，B-B B-C B-G 等信息、流向的变化，更像交易中心。

151、过滤器有哪些作用？【基础】

答：可以验证客户是否来自可信的网络，可以对客户提交的数据进行重新编码，可以从系统里获得配置的信息，可以过滤掉客户的某些不应该出现的词汇，可以验证用户是否登录，可以验证客户的浏览器是否支持当前的应用，可以记录系统的日志等等。

152、过滤器的用法？（对客户端的请求统一编码和对客户端进行认证）【基础】

答：首先要实现（implements）Filter 接口，同时覆盖 Filter 接口的三个方法：

```
init(FilterConfig config) //用于获得 FilterConfig 对象；
doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) //进行过滤处理一些业务；
destroy() //销毁 Filter。
```

153、简述 HttpSession 的作用、使用方法，可用代码说明。（3 分钟）【基础】

答：HttpSession 中可以跟踪并储存用户信息，把值设置到属性中，有 2 个方法：setAttribute(),getAttrribute()；

例如：在一个方法中用 session.setAttribute(“student”,student);在 session 中设置一个属性名为 student,值为一个名为 student 的对象。而后可在同一 session 范围内用 getAttribute(“student”)取出该属性，得到 student 对象。

154、介绍在 JSP 中如何使用 JavaBeans？【基础】

答：在 JSP 中使用 JavaBean 常用的动作有：

1）<jsp:useBean />：用来创建和查找 bean 对象；

2）<jsp:setProperty />：用来设置 bean 的属性，即调用其 setXxx()方法;

3）<jsp:getProperty />：用来获得 bean 的属性，即调用其 getXxx()方法。

155、JSP 和 Servlet 中的请求转发分别如何实现？【基础】
答：JSP 中的请求转发可利用 forward 动作实现：<jsp:forward />；
Serlvet 中实现请求转发的方式为：
getServletContext().getRequestDispatcher(path).forward(req,res)。

156、Web.Xml 的作用？【基础】
答：用于配置 web 应用的信息；如 listener、filter 及 servlet 的配置信息等。

157、写出熟悉的 JSTL 标签。【基础】
答：<c:if>、<c:choose>、<c: when>、<c: otherwise>、<c:forEach>、<c:set>。

158、说出 struts 中的标签。【基础】
答：<bean:message />　　<html:errors />
<bean:include />　　<html:messages />
<bean:define />　　<html:html>
<bean:write />　　<html:form>
<bean:resource />　　<html:link>
<bean:cokkie />　　<html:text>
<bean:heder />　　<logic:present />
<bean:parameter />　　<logic:equal />
<bean:size />

159、JSP 标签的作用？如何定义？【中等难度】
答：作用：分离 jsp 页面的内容和逻辑；
业务逻辑开发者可以创建自定义标签；
封装业务逻辑；
可重用并且易维护；
易于手工修改、易于工具维护；
提供简洁的语法；
定义:
写标签处理器；
写 tld 文件；
讲标签处理器和 tld 文件放到同一个包里面；
把 jsp 页面和标签库配置部署在一起。

160、写一个自定义标签；【中等难度】
答：代码如下：

```
import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;
public class TimeTag extends SimpleTagSupport{
    private boolean isServer = true;
    public void setServer(boolean isServer){
        this.isServer = isServer;
    }
```

```
        public void doTag() throws JspException, IOException{
            JspWriter out = getJspContext().getOut();
            if(isServer) {
                out.println(new java.util.Date());
            }else{
                out.println("<script language=\"javascript\">");
                out.println("document.write(new Date());");
                out.println("</script>");
            }
        }
    }
```

161、javascript 的优缺点和内置对象；【中等难度】
答：1）优点：简单易用，与 Java 有类似的语法，可以使用任何文本编辑工具编写，只需要浏览器就可执行程序，并且事先不用编译，逐行执行，无需进行严格的变量声明，而且内置大量现成对象，编写少量程序可以完成目标；
2）缺点：不适合开发大型应用程序；
3）Javascript 有 11 种内置对象：
Array、String、Date、Math、Boolean、Number、
Function、Global、Error、RegExp、Object。

七、EJB 及 Spring 部分：
（共 18 题：基础 4 道，中等难度 13 道，较难 1 道）

162、EJB 与 JAVA BEAN 的区别？【基础】
答：Java Bean 是可复用的组件，对 Java Bean 并没有严格的规范，理论上讲，任何一个 Java 类都可以是一个 Bean。但通常情况下，由于 Java Bean 是被容器所创建（如 Tomcat）的，所以 Java Bean 应具有一个无参的构造器，另外，通常 Java Bean 还要实现 Serializable 接口用于实现 Bean 的持久性。Java Bean 实际上相当于微软 COM 模型中的本地进程内 COM 组件，它是不能被跨进程访问的。Enterprise Java Bean 相当于 DCOM，即分布式组件。它是基于 Java 的远程方法调用（RMI）技术的，所以 EJB 可以被远程访问（跨进程、跨计算机）。但 EJB 必须被布署在诸如 Webspere、WebLogic 这样的容器中，EJB 客户从不直接访问真正的 EJB 组件，而是通过其容器访问。EJB 容器是 EJB 组件的代理，EJB 组件由容器所创建和管理。客户通过容器来访问真正的 EJB 组件。

163、EJB 的几种类型？【基础】
答：会话(Session)Bean、实体(Entity)Bean、消息驱动的(Message Driven)Bean; 会话 Bean 又可分为有状态（Stateful）和无状态（Stateless）两种；实体 Bean 可分为 Bean 管理的持续性（BMP）和容器管理的持续性（CMP）两种。

164、remote 接口和 home 接口主要作用？【基础】
答：remote 接口定义了业务方法，用于 EJB 客户端调用业务方法；
home 接口是 EJB 工厂用于创建和移除查找 EJB 实例。

165、客服端口调用 EJB 对象的几个基本步骤？【基础】
答：设置 JNDI 服务工厂以及 JNDI 服务地址系统属性，查找 Home 接口，从 Home 接口调用 Create 方法创建 Remote 接口，通过 Remote 接口调用其业务方法。

166、EJB 的角色和三个对象？【中等难度】
答：一个完整的基于 EJB 的分布式计算结构由六个角色组成，这六个角色可以由不同的开发商提供，每个角色所作的工作必须遵循 Sun 公司提供的 EJB 规范，以保证彼此之间的兼容性。这六个角色分别是 EJB 组件开发者（Enterprise Bean Provider） 、应用组合者（Application Assembler）、部署者（Deployer）、EJB 服务器提供者（EJB Server Provider）、EJB 容器提供者（EJB Container Provider）、系统管理员（System Administrator），这里面，EJB 容器是 EJB 之所以能够运行的核心。EJB 容器管理着 EJB 的创建，撤消，激活，去活，与数据库的连接等等重要的核心工作；三个对象是 Remote（Local）接口、Home（LocalHome）接口，Bean 类。

167、EJB 是基于哪些技术实现的？并说出 SessionBean 和 EntityBean 的区别，StatefulBean 和 StatelessBean 的区别。【中等难度】
答：EJB 包括 Session Bean、Entity Bean、Message Driven Bean，基于 JNDI、RMI、JTA 等技术实现。

SessionBean 在 J2EE 应用程序中被用来完成一些服务器端的业务操作，例如访问数据库、调用其他 EJB 组件。EntityBean 被用来代表应用系统中用到的数据。

对于客户机，SessionBean 是一种非持久性对象，它实现某些在服务器上运行的业务逻辑。

对于客户机，EntityBean 是一种持久性对象，它代表一个存储在持久性存储器中的实体的对象视图，或是一个由现有企业应用程序实现的实体。

Session Bean 还可以再细分为 Stateful Session Bean 与 Stateless Session Bean ，这两种的 Session Bean 都可以将系统逻辑放在 method 之中执行，不同的是 Stateful Session Bean 可以记录呼叫者的状态，因此通常来说，一个使用者会有一个相对应的 Stateful Session Bean 的实体。Stateless Session Bean 虽然也是逻辑组件，但是他却不负责记录使用者状态，也就是说当使用者呼叫 Stateless Session Bean 的时候，EJB Container 并不会找寻特定的 Stateless Session Bean 的实体来执行这个 method。换言之，很可能数个使用者在执行某个 Stateless Session Bean 的 methods 时，会是同一个 Bean 的 Instance 在执行。从内存方面来看， Stateful Session Bean 与 Stateless Session Bean 比较，Stateful Session Bean 会消耗 J2EE Server 较多的内存，然而 Stateful Session Bean 的优势却在于他可以维持使用者的状态。

168、bean 实例的生命周期？【中等难度】
答：对于 Stateless Session Bean、Entity Bean、Message Driven Bean 一般存在缓冲池管理，而对于 Entity Bean 和 Statefull Session Bean 存在 Cache 管理，通常包含创建实例，设置上下文、创建 EJB Object（create）、业务方法调用、remove 等过程，对于存在缓冲池管理的 Bean，在 create 之后实例并不

从内存清除，而是采用缓冲池调度机制不断重用实例，而对于存在 Cache 管理的 Bean 则通过激活和去激活机制保持 Bean 的状态并限制内存中实例数量。

169、EJB 的激活机制？【中等难度】
答：以 Stateful Session Bean 为例：其 Cache 大小决定了内存中可以同时存在的 Bean 实例的数量，根据 MRU 或 NRU 算法，实例在激活和去激活状态之间迁移，激活机制是当客户端调用某个 EJB 实例业务方法时，如果对应 EJB Object 发现自己没有绑定对应的 Bean 实例则从其去激活 Bean 存储中(通过序列化机制存储实例）回复（激活）此实例。状态变迁前会调用对应的 ejbActive 和 ejbPassivate 方法。

170、EJB 包括（SessionBean,EntityBean）说出他们的生命周期，及如何管理事务的？【中等难度】
答：SessionBean：Stateless Session Bean 的生命周期是由容器决定的，当客户机发出请求要建立一个 Bean 的实例时，EJB 容器不一定要创建一个新的 Bean 的实例供客户机调用，而是随便找一个现有的实例提供给客户机。当客户机第一次调用一个 Stateful Session Bean 时，容器必须立即在服务器中创建一个新的 Bean 实例,并关联到客户机上,以后此客户机调用 Stateful Session Bean 的方法时容器会把调用分派到与此客户机相关联的 Bean 实例。EntityBean: Entity Beans 能存活相对较长的时间，并且状态是持续的。只要数据库中的数据存在，Entity beans 就一直存活。而不是按照应用程序或者服务进程来说的。即使 EJB 容器崩溃了，Entity beans 也是存活的。Entity Beans 生命周期能够被容器或者 Beans 自己管理。EJB 通过以下技术管理事务：对象管理组织（OMG）的对象实务服务（OTS），Sun Microsystems 的 Transaction Service（JTS）、Java Transaction API（JTA），开发组（X/Open）的 XA 接口。

171、EJB 的事务是如何实现的？何时进行回滚；【中等难度】
答：是通过使用容器或 Bean 自身管理事务的；
　　当产生一个系统异常时容器就自动回滚事务。

172、EJB 容器提供的服务？【中等难度】
答：主要提供生命周期管理、代码产生、持续性管理、安全、事务管理、锁和并发行管理等服务。

173、EJB 需直接实现它的业务接口或 Home 接口吗？请简述理由。【中等难度】
答：远程接口和 Home 接口不需要直接实现，他们的实现代码是由服务器产生的，程序运行中对应实现类会作为对应接口类型的实例被使用。

174、请对以下在 J2EE 中常用的名词进行解释(或简单描述) 【中等难度】
答：web 容器：给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使 JSP,SERVLET 直接跟容器中的环境变量接口交互，不必关注其它系统问题。主要由 WEB 服务器来实现。例如：TOMCAT,WEBLOGIC,WEBSPHERE 等。该容器提供的接口严格遵守 J2EE 规范中的 WEB APPLICATION 标准。我们把遵守以上标准的 WEB 服务器就叫做 J2EE 中的 WEB 容器；
　　EJB 容器：Enterprise java bean 容器。更具有行业领域特色。他提供给

运行在其中的组件 EJB 各种管理功能。只要满足 J2EE 规范的 EJB 放入该容器，马上就会被容器进行高效率的管理。并且可以通过现成的接口来获得系统级别的服务。例如邮件服务、事务管理；

JNDI：（Java Naming & Directory Interface）JAVA 命名目录服务。主要提供的功能是：提供一个目录系统，让其它各地的应用程序在其上面留下自己的索引，从而满足快速查找和定位分布式应用程序的功能；

JMS：（Java Message Service）JAVA 消息服务。主要实现各个应用程序之间的通讯。包括点对点和广播；

JTA：（Java Transaction API）JAVA 事务服务。提供各种分布式事务服务。应用程序只需调用其提供的接口即可；

JAF：（Java Action FrameWork）JAVA 安全认证框架。提供一些安全控制方面的框架。让开发者通过各种部署和自定义实现自己的个性安全控制策略；

RMI/IIOP:（Remote Method Invocation /internet 对象请求中介协议）他们主要用于通过远程调用服务。例如，远程有一台计算机上运行一个程序，它提供股票分析服务，我们可以在本地计算机上实现对其直接调用。当然这是要通过一定的规范才能在异构的系统之间进行通信。RMI 是 JAVA 特有的。

175、J2EE 是什么？【中等难度】

答：J2EE 是 Sun 公司提出的多层(multi-diered),分布式(distributed),基于组件(component-base)的企业级应用模型(enterpriese application model).在这样的一个应用系统中，可按照功能划分为不同的组件，这些组件又可在不同计算机上，并且处于相应的层次(tier)中。所属层次包括客户层(clietn tier)组件,web 层和组件,Business 层和组件,企业信息系统(EIS)层。

176、J2EE 是技术还是平台还是框架？【中等难度】

答：J2EE 本身是一个标准，一个为企业分布式应用的开发提供的标准平台；

J2EE 也是一个框架，包括 JDBC、JNDI、RMI、JMS、EJB、JTA 等技术。

177、请写出 spring 中 IOC 的三种实现机制。【中等难度】

答：三种机制为：通过 setter 方法注入、通过构造方法注入和接口注入。

178、写出你熟悉的开源框架以及各自的作用。【中等难度】

答：框架：hibernate、spring、struts；

Hibernate 主要用于数据持久化；

Spring 的控制反转能起到解耦合的作用；

Struts 主要用于流程控制。

179、EJB 规范规定 EJB 中禁止的操作有哪些？【较难】

答：1)不能操作线程和线程 API(线程 API 指非线程对象的方法，如 notify,wait 等)；

2)不能操作 awt；

3)不能实现服务器功能；

4)不能对静态属性存取；

5)不能使用 IO 操作直接存取文件系统；

6)不能加载本地库；

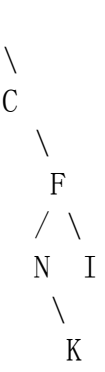
7)不能将 this 作为变量和返回；
8)不能循环调用。

八、数据结构、算法及计算机基础部分：
（共 8 题：基础 6 道，中等难度 1 道，较难 1 道）

180、一个 byte 几个单位？【基础】
答：8bit。

181、常用 UNIX 命令(Linux 的常用命令)（至少 10 个）【基础】
答：ls pwd mkdir rm cp mv cd ps ftp telnet ping env more echo

182、后序遍历下列二叉树，访问结点的顺序是？【基础】



答：顺序为：DJGEBKNIFCA 。

183、排序都有哪几种方法？请列举。用 JAVA 实现一个快速排序。【基础】
答：排序的方法有：插入排序（直接插入排序、希尔排序），交换排序（冒泡排序、快速排序），选择排序（直接选择排序、堆排序），归并排序，分配排序（箱排序、基数排序）；
快速排序的伪代码：
//使用快速排序方法对 a[0 :n- 1]排序
从 a[0 :n- 1]中选择一个元素作为 middle，该元素为支点；
把余下的元素分割为两段 left 和 right，使得 left 中的元素都小于等于支点，而 right 中的元素都大于等于支点；
递归地使用快速排序方法对 left 进行排序；
递归地使用快速排序方法对 right 进行排序；
所得结果为 left + middle + right。

184、写一种常见排序。【基础】
答：C++中冒泡排序：

```
void swap( int& a, int& b ){
   int c=a; a = b; b = c;
}
void bubble( int* p, int len ){
   bool bSwapped;
```

```
    do {
        bSwapped = false;
        for( int i=1; i<len; i++ ){
            if( p[i-1]>p[i] ){
                swap( p[i-1], p[i] );
                bSwapped = true;
            }
        }
    }while( bSwapped );
}
```

185、写一个一小段程序检查数字是否为质数；以上的程序你采用的哪种语言写的？采用该种语言的理由是什么？【基础】

答：代码如下：

```
#include <math.h>
bool prime( int n ){
    if(n<=0) exit(0);
    for( int i=2; i<=n; i++ )
        for( int j=2; j<=sqrt(i); j++)
            if((n%j==0) && (j!=n))
                return false;
    return true;
}
```

采用 C++，因为其运行效率高。

186、编程题：设有 n 个人依围成一圈，从第 1 个人开始报数，数到第 m 个人出列，然后从出列的下一个人开始报数，数到第 m 个人又出列，…，如此反复到所有的人全部出列为止。设 n 个人的编号分别为 1，2，…，n，打印出出列的顺序；要求用 java 实现。【中等难度】

答：代码如下：

```
package test;
public class CountGame {
    private static boolean same(int[] p,int l,int n){
        for(int i=0;i<l;i++){
            if(p[i]==n){
                return true;
            }
        }
        return false;
    }
    public static void play(int playerNum, int step){
        int[] p=new int[playerNum];
        int counter = 1;
        while(true){
```

```
                if(counter > playerNum*step){
                    break;
                }
                for(int i=1;i<playerNum+1;i++){
                    while(true){
                        if(same(p,playerNum,i)==false) break;
                        else i=i+1;
                    }
                    if(i > playerNum)break;
                    if(counter%step==0){
                        System.out.print(i + " ");
                        p[counter/step-1]=i;
                    }
                    counter+=1;
                }
            }
           System.out.println();
        }
        public static void main(String[] args) {
            play(10, 7);
        }
    }
```

187、写一个方法 1000 的阶乘。【较难】

答：C++的代码实现如下：

```
    #include <iostream>
    #include <iomanip>
    #include <vector>
    using namespace std;
    class longint {
    private:
        vector<int> iv;
    public:
        longint(void) { iv.push_back(1); }
        longint& multiply(const int &);
        friend ostream& operator<<(ostream &, const longint &);
    };
    ostream& operator<<(ostream &os, const longint &v) {
        vector<int>::const_reverse_iterator iv_iter = v.iv.rbegin();
        os << *iv_iter++;
        for ( ; iv_iter < v.iv.rend(); ++iv_iter) {
            os << setfill('0') << setw(4) << *iv_iter;
        }
        return os;
```

```
}
longint& longint::multiply(const int &rv) {
    vector<int>::iterator iv_iter = iv.begin();
    int overflow = 0, product = 0;
    for ( ; iv_iter < iv.end(); ++iv_iter) {
        product = (*iv_iter) * rv;
        product += overflow;
        overflow = 0;
        if (product > 10000) {
            overflow = product / 10000;
            product -= overflow * 10000;
        }
        iv_iter = product;
    }
    if (0 != overflow) {
        iv.push_back(overflow);
    }
    return *this;
}
int main(int argc, char **argv) {
    longint result;
    int l = 0;
    if(argc==1){
        cout << "like: multiply 1000" << endl;
        exit(0);
    }
    sscanf(argv[1], "%d", &l);
    for (int i = 2; i <= l; ++i) {
        result.multiply(i);
    }
    cout << result << endl;
    return 0;
}
```

九、C++部分：（共 14 题：基础 10 道，中等 1 道，较难 3 道）

188、以下三条输出语句分别输出什么？【基础】

```
char str1[] = "abc";
char str2[] = "abc";
const char str3[] = "abc";
const char str4[] = "abc";
const char* str5 = "abc";
const char* str6 = "abc";
```

```
cout << boolalpha << (str1==str2) << endl; //输出什么?
cout << boolalpha << (str3==str4) << endl; //输出什么?
cout << boolalpha << (str5==str6) << endl; //输出什么?
```
答：输出为：false、false、true。

189、以下反向遍历 array 数组的方法有什么错误？【基础】
```
vector<int> array;
array.push_back(1);
array.push_back(2);
array.push_back(3);
//反向遍历 array 数组：
for(vector<int>::size_type i=array.size()-1; i>=0; --i){
    cout << array[i] << endl;
}
```
答：for 循环中的变量 i 的类型不应定义为 vector<int>::size_type，因为该类型为无符号数值类型，故循环条件将恒成立，为死循环，应将其类型定义为有符号的 int 类型。

190、以下代码有什么问题？【基础】
```
cout << (true ? 1 : "1") << endl;
```
答：运算符中两个可选值的类型不同。

191、以下代码有什么问题？【基础】
```
typedef vector<int> IntArray;
IntArray array;
array.push_back(1);
array.push_back(2);
array.push_back(2);
array.push_back(3);
//删除 array 数组中所有的 2
for(IntArray::iterator itor=array.begin(); itor!=array.end();
                                                    ++itor){
    if(2==*itor) {
        array.erase(itor);
    }
}
```
答：for 循环中的 if 语句后的 array.erase(itor)语句，它将迭代器 itor 所指向的元素删除后会自动下移一位，故应在其后加上语句：itor--;

192、以下代码中的两个 sizeof 用法有问题吗？【基础】
```
void upperCase(char str[]){ //将 str 中的小写字母转换成大写字母
    for(int i=0; i<sizeof(str)/sizeof(str[0]); ++i){
        if('a'<=str[i] && str[i]<='z')
            str[i] -= ('a'-'A');
    }
```

```
}
int main(){
    char str[] = "aBcDe";
    cout << "str 字符串长度为:" << sizeof(str)/sizeof(str[0]);
    cout << endl;
    upperCase(str);
    cout << str << endl;
    return 0;
}
```

答：在 upperCase 方法中，for 循环的 sizeof(str)的值将总是 4，所以该方法只能将参数中的字符串的前四个字符转换成大写字母。

193、以下代码能够编译通过吗？为什么？【基础】

```
    unsigned int const size1 = 2;
    char str1[size1];
    unsigned int temp = 0;
    cin >> temp;
    unsigned int const size2 = temp;
    char str2[size2];
```

答：能；

194、以下代码有什么问题？【基础】

```
struct Test{
    Test(int){}
    Test(){}
    void fun(){}
};
void main(void){
    Test a(1);
    a.fun();
    Test b();
    b.fun();
}
```

答：main 函数的返回类型应为 int；不能对 b 调用 fun（）方法。

195、以下代码中的输出语句输出 0 吗？为什么？【基础】

```
    struct CLS{
        int m_i;
        CLS(int i):m_i(i){ }
        CLS(){ CLS(0);}
    };
    int main(){
        CLS obj;
        cout <<obj.m_i << endl;
```

```
}
```
答：输出不是 0；

196、C++中的空类，默认产生哪些类成员函数？【基础】
答：空类中默认包含的成员函数如下：
```
class Empty{
public:
    Empty(); //缺省构造函数
    Empty( const Empty& ); //拷贝构造函数
    ~Empty(); //析构函数
    Empty& operator=( const Empty& ); //赋值运算符
    Empty* operator&(); //取址运算符
    const Empty* operator&() const; //取址运算符 const
};
```

197、统计一篇文章中单词个数。【基础】
答：代码如下：
```
include<iostream>
#include<fstream>
using namespace std;
int main(){
    ifstream fin("t.txt");
    if(!fin){
        cout<<"can't open file"<<endl;
        return -1;
    }
    int count = 0;
    char buf[256];
    memset(buf, 0, 256);
    while(1){
        fin2>>buf;
        if(fin2.eof())
        break;
        count++;
    }
    cout<<"The number of the words is : "<<count<<endl;
    fin2.close();
    return 0;
}
```

198、写一个函数，完成内存之间的拷贝。【中等难度】
答：代码如下：
```
void* mymemcpy(void* dest, const void* src, size_t count){
    char* pdest = static_cast<char*>(dest);
    const char* psrc = static_cast<const char*>(src);
```

```
        if(pdest>psrc && pdest<psrc+count){ //能考虑到这种情况就行了
            for(size_t i=count-1; i!=-1; --i){
                pdest[i] = psrc[i];
            }
        }else{
            for(size_t i=0; i<count; ++i){
                pdest[i] = psrc[i];
            }
        }
        return dest;
    }
    int main(){
        char str[] = "0123456789";
        mymemcpy(str+1, str+0, 9);
        cout << str << endl; //将输出"0012345678"
        return 0;
    }
```

199、非 C++内建类型 A 和 B，在哪几种情况下 B 能隐式转化为 A？【较难】

答：a）class B : public A{……}//B 公有继承自 A，可以是间接继承的

b）class B{operator A();}//B 实现了隐式转化为 A 的转化

c）class A{ A(const B&);}//A 实现了 non-explicit 的参数为 B 构造函数（可以有其他带带默认值的参数）

d）A& operator= (const A&);//赋值操作，虽不是正宗的隐式类型转换，但也可以勉强算一个

200、以下代码有什么问题？【较难】

```
    void char2Hex(char c){ //将字符以 16 进制显示
        char ch = c/0x10 + '0';
        if(ch>'9') ch += ('A'-'9'-1);
        char cl = c%0x10 + '0';
        if(cl>'9') cl += ('A'-'9'-1);
        cout << ch << cl << ' ';
    }
    int main(){
        char str[] = "I love 中国";
        for(size_t i=0; i<strlen(str); ++i)
            char2Hex(str[i]);
        cout << endl;
        return 0;
    }
```

答：

201、以下两条输出语句分别输出什么？【较难】

```
float a = 1.0f;
```

```
cout << (int)a << endl;
cout << (int&)a << endl;
cout << boolalpha << ((int)a==(int&)a) << endl; //输出什么
float b = 0.0f;
cout << (int)b << endl;
cout << (int&)b << endl;
cout << boolalpha << ((int)b==(int&)b) << endl;//输出什么
```

答：第一处输出 false，第二处输出 true。

十、WebLogic 及其它：（共 13 题：附加部分，超出授课范围）

1、如何给 weblogic 指定大小的内存?
答：在启动 Weblogic 的脚本中（位于所在 Domian 对应服务器目录下的 startServerName），增加 set MEM_ARGS=-Xms32m -Xmx200m，可以调整最小内存为 32M，最大 200M。

2、如何设定的 weblogic 的热启动模式(开发模式)与产品发布模式?
答：可以在管理控制台中修改对应服务器的启动模式为开发或产品模式之一，或者修改服务的启动文件或者 commenv 文件，增加 set PRODUCTION_MODE=true。

3、如何启动时不需输入用户名与密码?
答:修改服务启动文件,增加 WLS_USER 和 WLS_PW 项;也可以在 boot.properties 文件中增加加密过的用户名和密码。

4、在 weblogic 管理控制台中对一个应用域(或者说是一个网站,Domain)进行 jms 及 ejb 或连接池等相关信息进行配置后,实际保存在什么文件中?
答：保存在此 Domain 的 config.xml 文件中，它是服务器的核心配置文件。

5、说说 weblogic 中一个 Domain 的缺省目录结构?比如要将一个简单的 helloWorld.jsp 放入何目录下,然后在浏览器上打入 http://主机:端口号 //helloword.jsp 就可以看到运行结果了？又比如这其中用到了一个自己写的 javaBean 该如何办?
答：Domain 目录\服务器目录\applications，将应用目录放在此目录下将可以作为应用访问，如果是 Web 应用，应用目录需要满足 Web 应用目录要求，jsp 文件可以直接放在应用目录中，Javabean 需要放在应用目录的 WEB-INF 目录的 classes 目录中，设置服务器的缺省应用将可以实现在浏览器上无需输入应用名。

6、在 weblogic 中发布 ejb 需涉及到哪些配置文件？
答：不同类型的 EJB 涉及的配置文件不同，都涉及到的配置文件包括 ejb-jar.xml,weblogic-ejb-jar.xml，CMP 实体 Bean 一般还需要 weblogic-cmp-rdbms-jar.xml

7、如何在 weblogic 中进行 ssl 配置与客户端的认证配置或说说 j2ee(标准)进行 ssl 的配置？

答：缺省安装中使用 DemoIdentity.jks 和 DemoTrust.jks　KeyStore 实现 SSL，需要配置服务器使用 Enable SSL，配置其端口，在产品模式下需要从 CA 获取私有密钥和数字证书，创建 identity 和 trust keystore，装载获得的密钥和数字证书。可以配置此 SSL 连接是单向还是双向的。

8、如何查看在 weblogic 中已经发布的 EJB?
答：可以使用管理控制台，在它的 Deployment 中可以查看所有已发布的 EJB。

9、CORBA 是什么?用途是什么?
答：CORBA 标准是公共对象请求代理结构(Common Object Request Broker Architecture)，由对象管理组织 (Object Management Group，缩写为 OMG)标准化。它的组成是接口定义语言(IDL)，语言绑定(binding:也译为联编)和允许应用程序间互操作的协议。 其目的为：用不同的程序设计语言书写在不同的进程中运行，为不同的操作系统开发。

10、在 weblogic 中开发消息 Bean 时的 persistent 与 non-persisten 的差别?
答：persistent 方式的 MDB 可以保证消息传递的可靠性,也就是如果 EJB 容器出现问题而 JMS 服务器依然会将消息在此 MDB 可用的时候发送过来，而 non－persistent 方式的消息将被丢弃。

11、LINUX 下线程，GDI 类的解释？
答：LINUX 实现的就是基于核心轻量级进程的"一对一"线程模型，一个线程实体对应一个核心轻量级进程，而线程之间的管理在核外函数库中实现；GDI 类为图像设备编程接口类库。

12、Jdo 是什么?
答：JDO 是 Java 对象持久化的新的规范，为 java data object 的简称,也是一个用于存取某种数据仓库中的对象的标准化 API。JDO 提供了透明的对象存储，因此对开发人员来说，存储数据对象完全不需要额外的代码（如 JDBC API 的使用）。这些繁琐的例行工作已经转移到 JDO 产品提供商身上，使开发人员解脱出来，从而集中时间和精力在业务逻辑上。另外，JDO 很灵活，因为它可以在任何数据底层上运行。JDBC 只是面向关系数据库（RDBMS）JDO 更通用，提供到任何数据底层的存储功能，比如关系数据库、文件、XML 以及对象数据库（ODBMS）等等，使得应用可移植性更强。

13、WEB SERVICE 名词解释；JAXP、JAXM 的解释；SOAP、UDDI,WSDL 解释？
答：Web Service 是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得 Web Service 能与其他兼容的组件进行互操作；

JAXP(Java API for XML Parsing)定义了在 Java 中使用 DOM, SAX, XSLT 的通用的接口，这样在你的程序中你只要使用这些通用的接口，当你需要改变具体的实现时候也不需要修改代码；

JAXM(Java API for XML Messaging)是为 SOAP 通信提供访问方法和传输机制的 API；

WSDL 是一种 XML 格式，用于将网络服务描述为一组端点，这些端点对包含

面向文档信息或面向过程信息的消息进行操作。这种格式首先对操作和消息进行抽象描述，然后将其绑定到具体的网络协议和消息格式上以定义端点。相关的具体端点即组合成为抽象端点（服务）；

SOAP 即简单对象访问协议(Simple Object Access Protocol)，它是用于交换 XML 编码信息的轻量级协议；

UDDI 的目的是为电子商务建立标准；UDDI 是一套基于 Web 的、分布式的、为 Web Service 提供的、信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web Service 注册，以使别的企业能够发现的访问协议的实现标准。