1、面向对象的特征有哪些方面?

答：面向对象的特征主要有以下几个方面：

1)抽象：抽象是将一类对象的共同特征总结出来构造类的过程，包括数据抽象和行为抽象两方面。抽象只关注对象有哪些属性和行为，并不关注这些行为的细节是什么。

2)继承：继承是从已有类得到继承信息创建新类的过程。提供继承信息的类被称为父类（超类、基类）；得到继承信息的类被称为子类（派生类）。继承让变化中的软件系统有了一定的延续性，同时继承也是封装程序中可变因素的重要手段（如果不能理解请阅读阎宏博士的《Java 与模式》或《设计模式精解》中关于桥梁模式的部分）。

3)封装：通常认为封装是把数据和操作数据的方法绑定起来，对数据的访问只能通过已定义的接口。面向对象的本质就是将现实世界描绘成一系列完全自治、封闭的对象。我们在类中编写的方法就是对实现细节的一种封装；我们编写一个类就是对数据和数据操作的封装。可以说，封装就是隐藏一切可隐藏的东西，只向外界提供最简单的编程接口（可以想想普通洗衣机和全自动洗衣机的差别，明显全自动洗衣机封装更好因此操作起来更简单；我们现在使用的智能手机也是封装得足够好的，因为几个按键就搞定了所有的事情）。

4)多态性：多态性是指允许不同子类型的对象对同一消息作出不同的响应。简单的说就是用同样的对象引用调用同样的方法但是做了不同的事情。多态性分为编译时的多态性和运行时的多态性。如果将对象的方法视为对象向外界提供的服务，那么运行时的多态性可以解释为：当 A 系统访问 B 系统提供的服务时，B 系统有多种提供服务的方式，但一切对 A 系统来说都是透明的（就像电动剃须刀是 A 系统，它的供电系统是 B 系统，B 系统可以使用电池供电或者用交流电，甚至还有可能是太阳能，A 系统只会通过 B 类对象调用供电的方法，但并不知道供电系统的底层实现是什么，究竟通过何种方式获得了

动力)。方法重载（overload）实现的是编译时的多态性（也称为前绑定），而方法重写（override）实现的是运行时的多态性（也称为后绑定)。运行时的多态是面向对象最精髓的东西，要实现多态需要做两件事：1. 方法重写（子类继承父类并重写父类中已有的或抽象的方法)；2. 对象造型（用父类型引用引用子类型对象，这样同样的引用调用同样的方法就会根据子类对象的不同而表现出不同的行为)。

2、作用域 public,private,protected,以及不写时的区别

答：区别如下：

作用域	当前类	同包	子类	其他
public	√	√	√	√
protected	√	√	√	×
default	√	√	×	×
private	√	×	×	×

类的成员不写访问修饰时默认为 default。默认对于同一个包中的其他类相当于公开（public)，对于不是同一个包中的其他类相当于私有 (private)。受保护 (protected) 对子类相当于公开，对不是同一包中的没有父子关系的类相当于私有。

3、&和&&的区别

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）

4、String 是最基本的数据类型吗?

答：不是。Java 中的基本数据类型只有 8 个：byte、short、int、long、float、double、char、boolean；除了基本类型（primitive type）和枚举类型（enumeration type），剩下的都是引用类型（reference type）。 String 是一个类，属于是引用数据类型，因此说 String 不是基本的数据类型

5、float f=3.4;是否正确?

答:不正确。3.4 是双精度数，将双精度型（double）赋值给浮点型（float）属于下转型（down-casting，也称为窄化）会造成精度损失，因此需要强制类型转换 float f =(float)3.4; 或者写成 float f =3.4F;。

6、short s1 = 1; s1 = s1 + 1;有错吗?short s1 = 1; s1 += 1;有错吗?

答：对于 short s1 = 1; s1 = s1 + 1;由于 1 是 int 类型，因此 s1+1 运算结果也是 int 型，需要强制转换类型才能赋值给 short 型。而 short s1 = 1; s1 += 1;可以正确编译，因为 s1+= 1;相当于 s1 = (short)(s1 + 1);其中有隐含的强制类型转换。

7、Java 有没有 goto?

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?

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、数组的创建方式有哪几种？

数据类型 [] 数组名=new 数据类型[元素的个数]
数据类型 [] 数组名={元素值，元素值，元组值 3.....}
数据类型[] 数组名=new 数据类型{元素值，元素值，元素值.....}

10、简述逻辑操作(&,|,^)与条件操作(&&,||)的区别？

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、什么时候用 assert

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。在实现中，assertion 就是在程序中的一条语句，它对一个 boolean 表达式进行检查，一个正确程序必须保证这个 boolean 表达式的值为 true；如果该值为 false，说明程序已经处于不正确的状态下，系统将给出警告或退出。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

12、heap 和 stack 有什么区别？

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

13、Math.round(11.5) 等于多少？Math.round(-11.5)等于多少

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

14、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

答：早期的 JDK 中，switch（expr）中，expr 可以是 byte、short、char、int。从 1.5 版开始，Java 中引入了枚举类型（enum），expr 也可以是枚举，从 JDK 1.7 版开始，还可以是字符串（String）。长整型（long）是不可以的。

15、如何快速的把 2 计算成 8

答： 2<<3

16、数组有没有 length()方法?String 有没有 length()方法

答：数组没有 length()这个方法，有 length 的属性。String 有 length()这个方法

17、在 Java 中，如何跳出当前的多重嵌套循环？

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。

18、构造器（constructor）是否可被重写（override）

答：构造器不能被继承，因此不能被重写，但可以被重载。

19、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？

不对，有相同的 hash code

20、自己定的类是否可以继承 String 类？

答：String 类是 final 类，不可以被继承

21、以下二条语句返回值为 true 的有：

A：“beijing”==“beijing”；

B：“beijing”.equalsIgnoreCase（new String（“beijing”））

答：A 和 B

22、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?

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

23、String 和 StringBuffer 的区别?

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

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?

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

```
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?

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?

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?

答：声明方法的存在而不去实现它的类被叫做抽象类（abstract class），。然而可以创建一个变量，其类型是一个抽象类，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 publicstatic 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?

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

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?

答：都不能。

33、静态变量和实例变量的区别？

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

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？

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?

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。

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？

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”);创建了几个对象?

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)？

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

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？

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

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： 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)？

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

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？

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？

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?

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：

```
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?

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?

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

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

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？

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

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”

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？

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

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 中的异常处理机制的简单原理和应用？

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。

61、error 和 exception 有什么区别?

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。

62、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?

答：会执行，在 return 前执行。

63、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

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。

64、运行时异常与一般异常有何异同？

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。

65、给我一个你最常见到的 runtime exception

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

66、final，finally，finalize 的区别

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。

67、类 Example A 继承 Exception，类 ExampleB 继承 Example A

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。

68、介绍 JAVA 中的 Collection FrameWork(及如何写自己的数据结构)

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。

69、List,Set,Map 是否继承自 Collection 接口

答：List,Set 是；Map 不是。

70、你所知道的集合类都有哪些？主要方法

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。

71、说出 ArrayList,Vector, LinkedList 的存储性能和特性

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。

72、Collection 和 Collections 的区别

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。

73、HashMap 和 Hashtable 的区别

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。

74、Arraylist 与 Vector 区别

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。

75、List、Map、Set 三个接口，存取元素时，各有什么特点

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。

76、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？

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。

77、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

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();
}
}
```

78、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？

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));
}
}
}
```

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。

79、sleep() 和 wait() 有什么区别?

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。

80、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

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。

81、请说出你所知道的线程同步的方法

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。

82、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?

答：1、继承 Thread 类实现多线程

2、实现 Runnable 接口方式实现多线程

3、使用 ExecutorService、Callable、Future 实现有返回结果的多线程，分别是 synchronized,wait 与 notify。

83、同步和异步有何异同，在什么情况下分别使用他们?

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。

84、启动一个线程是用 run()还是 start()?

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。

85、线程的基本概念、线程的基本状态以及状态之间的关系

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。

86、简述 synchronized 和 java.util.concurrent.locks.Lock 的异同？

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。

87、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？

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。

88、一个线程运行时发生异常会怎样

如果异常没有被捕获该线程将会停止执行。Thread.UncaughtExceptionHandler 是用于处理未捕获异常造成线程突然中断情况的一个内嵌接口。当一个未捕获异常将造成线程中断的时候 JVM 会使用 Thread.getUncaughtExceptionHandler()来查询线程的 UncaughtExceptionHandler 并将线程和异常作为参数传递给 handler 的 uncaughtException()方法进行处理。

89、线程 yield()方法有什么用

Yield 方法可以暂停当前正在执行的线程对象，让其它有相同优先级的线程执行。它是一个静态方法而且只保证当前线程放弃 CPU 占用而不能保证使其它线程一定能占用 CPU，执行 yield()的线程有可能在进入到暂停状态后马上又被执行

90、什么是重入锁

所谓重入锁，指的是以线程为单位，当一个线程获取对象锁之后，这个线程可以再次获取本对象上的锁，而其他的线程是不可以的。

91、线程数过多会造成什么异常？

线程过多会造成栈溢出，也有可能会造成堆异常

92、什么是 CAS 算法？在多线程中有哪些应用

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

93、线程同步需要注意什么

1、尽量缩小同步的范围，增加系统吞吐量。
2、分布式同步锁无意义，要使用分布式锁。
3、防止死锁，注意加锁顺序。

94、线程之间如何传递数据

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

95 说几个常用的 Lock 接口实现锁

ReentrantLock、ReadWriteLock

96、什么是乐观锁和悲观锁？

乐观锁：就像它的名字一样，对于并发间操作产生的线程安全问题持乐观状态，乐观锁认为竞争不总是会发生，因此它不需要持有锁，将比较-替换这两个动作作为一个原子操作尝试去修改内存中的变量，如果失败则表示发生冲突，那么就应该有相应的重试逻辑。

悲观锁：还是像它的名字一样，对于并发间操作产生的线程安全问题持悲观状态，悲观锁认为竞争总是会发生，因此每次对某资源进行操作时，都会持有一个独占的锁，就像 synchronized，不管三七二十一，直接上了锁就操作资源了

97、Hashtable 的 size()方法为什么要做同步？

同一时间只能有一条线程执行固定类的同步方法，但是对于类的非同步方法，可以多条线程同时访问。所以，这样就有问题了，可能线程 A 在执行 Hashtable 的 put 方法添加数据，线程 B 则可以正常调用 size()方法读取 Hashtable 中当前元素的个数，那读取到的值可能不是最新的，可能线程 A 添加了完了数据，但是没有对 size++，线程 B 就已经读取 size 了，那么对于线程 B 来说读取到的 size 一定是不准确的。而给 size()方法加了同步之后，意味着线程 B 调用 size()方法只有在线程 A 调用 put 方法完毕之后才可以调用，这样就保证了线程安全性

CPU 执行代码，执行的不是 Java 代码，这点很关键，一定得记住。Java 代码最终是被翻译成机器码执行的，机器码才是真正可以和硬件电路交互的代码。即使你看到 Java 代码只有一行，甚至你看到 Java 代码编译之后生成的字节码也只有一行，也不意味着对于底层来说这句语句的操作只有一个。一句"return count"假设被翻译成了三句汇编语句执行，一句汇编语句和其机器码做对应，完全可能执行完第一句，线程就切换了。

98、什么是自旋锁

自旋锁是采用让当前线程不停地的在循环体内执行实现的，当循环的条件被其他线程改变时才能进入临界区

99、什么是阻塞式方法

阻塞式方法是指程序会一直等待该方法完成期间不做其他事情，ServerSocket 的 accept()方法就是一直等待客户端连接。这里的阻塞是指调用结果返回之前，当前线程会被挂起，直到得到结果之后才会返回。此外，还有异步和非阻塞式方法在任务完成前就返回。

100、提交任务时线程池队列已满会时发会生什么？

当线程数小于最大线程池数 maximumPoolSize 时就会创建新线程来处理，而线程数大于等于最大线程池数 maximumPoolSize 时就会执行拒绝策略。

101、java 多线程中的死锁、活锁、饥饿、无锁都是什么鬼？

死锁、活锁、饥饿是关于多线程是否活跃出现的运行阻塞障碍问题，如果线程出现了这三种情况，即线程不再活跃，不能再正常地执行下去了。

死锁

死锁是多线程中最差的一种情况，多个线程相互占用对方的资源的锁，而又相互等对方释放锁，此时若无外力干预，这些线程则一直处理阻塞的假死状态，形成死锁。

举个例子，A 同学抢了 B 同学的钢笔，B 同学抢了 A 同学的书，两个人都相互占用对方的东西，都在让对方先还给自己自己再还，这样一直争执下去等待对方还而又得不到解决，老师知道此事后就让他们相互还给对方，这样在外力的干预下他们才解决，当然这只是个例子没有老师他们也能很好解决，计算机不像人如果发现这种情况没有外力干预还是会一直阻塞下去的。

活锁

活锁这个概念大家应该很少有人听说或理解它的概念，而在多线程中这确实存在。活锁恰恰与死锁相反，死锁是大家都拿不到资源都占用着对方的资源，而活锁是拿到资源却又相互释放不执行。当多线程中出现了相互谦让，都主动将资源释放给别的线程使用，这样这个资源在多个线程之间跳动而又得不到执行，这就是活锁。

饥饿

我们知道多线程执行中有线程优先级这个东西，优先级高的线程能够插队并优先执行，这样如果优先级高的线程一直抢占优先级低线程的资源，导致低优先级线程无法得到执行，这就是饥饿。当然还有一种饥饿的情况，一个线程一直占着一个资源不放而导致其他线程得不到执行，与死锁不同的是饥饿在以后一段时间内还是能够得到执行的，如那个占用资源的线程结束了并释放了资源。

无锁

无锁，即没有对资源进行锁定，即所有的线程都能访问并修改同一个资源，但同时只有一个线程能修改成功。无锁典型的特点就是一个修改操作在一个循环内进行，线程会不断的尝试修改共享资源，如果没有冲突就修改成功并退出否则就会继续下一次循环尝试。所以，如果有多个线程修改同一个值必定会有一个线程能修改成功，而其他修改失败的线程会不断重试直到修改成功。之前的文章我介绍过 JDK 的 CAS 原理及应用即是无锁的实现。

可以看出，无锁是一种非常良好的设计，它不会出现线程出现的跳跃性问题，锁使用不当肯定会出现系统性能问题，虽然无锁无法全面代替有锁，但无锁在某些场合下是非常高效的。

102、什么是原子性、可见性、有序性

原子性、可见性、有序性是多线程编程中最重要的几个知识点，由于多线程情况复杂，如何让每个线程能看到正确的结果，这是非常重要的。

原子性

原子性是指一个线程的操作是不能被其他线程打断，同一时间只有一个线程对一个变量进行操作。在多线程情况下，每个线程的执行结果不受其他线程的干扰，比如说多个线程同时对同一个共享成员变量 n++100 次，如果 n 初始值为 0，n 最后的值应该是 100，所以说它们是互不干扰的，这就是传说的中的原子性。但 n++并不是原子性的操作，要使用 AtomicInteger 保证原子性。

可见性

可见性是指某个线程修改了某一个共享变量的值，而其他线程是否可以看见该共享变量修改后的值。在单线程中肯定不会有这种问题，单线程读到的肯定都是最新的值，而在多线程编程中就不一定了。

每个线程都有自己的工作内存，线程先把共享变量的值从主内存读到工作内存，形成一个副本，当计算完后再把副本的值刷回主内存，从读取到最后刷回主内存这是一个过程，当还没刷回主内存的时候这时候对其他线程是不可见的，所以其他线程从主内存读到的值是修改之前的旧值。

像 CPU 的缓存优化、硬件优化、指令重排及对 JVM 编译器的优化，都会出现可见性的问题。

有序性

我们都知道程序是按代码顺序执行的，对于单线程来说确实是如此，但在多线程情况下就不是如此了。为了优化程序执行和提高 CPU 的处理性能，JVM 和操作系统都会对指令进行重排，也就说前面的代码并不一定都会在后面的代码前面执行，即后面的代码可能会插到前面的代码之前执行，只要不影响当前线程的执行结果。所以，指令重排只会保证当前线程执行结果一致，但指令重排后势必会影响多线程的执行结果。

103、线程池的作用

线程池作用就是限制系统中执行线程的数量。

根据系统的环境情况，可以自动或手动设置线程数量，达到运行的最佳效果；少了浪费了系统资源，多了造成系统拥挤效率不高。用线程池控制线程数量，其他线程排队等候。一个任务执行完毕，再从队列的中取最前面的任务开始执行。若队列中没有等待进程，线程池的这一资源处于等待。当一个新任务需要运行时，如果线程池中有等待的工作线程，就可以开始运行了；否则进入等待队列。

105、什么是 Java 虚拟机

Java 虚拟机是一个想象中的机器,在实际的计算机上通过软件模拟来实现。Java 虚拟机有自己想象中的硬件,如处理器、堆栈、寄存器等,还具有相应的指令系统。

106、为什么使用 Java 虚拟机

Java 语言的一个非常重要的特点就是与平台的无关性。而使用 Java 虚拟机是实现这一特点的关键。一般的高级语言如果要在不同的平台上运行,至少需要编译成不同的目标代码。而引入 Java 语言虚拟机后,Java 语言在不同平台上运行时不需要重新编译。Java 虚拟机屏蔽了与具体平台相关的信息,使得 Java 语言编译程序只需生成在 Java 虚拟机上运行的目标代码(字节码),就可以在多种平台上不加修改地运行。Java 虚拟机在执行字节码时,把字节码解释成具体平台上的机器指令执行。

107、Java 虚拟机的生命周期

一个运行中的 Java 虚拟机有着一个清晰的任务：执行 Java 程序。程序开始执行时它才运行，程序结束时它就停止。假如你同时运行三个 Java 程序，就会有三个运行中的 Java 虚拟机。

Java 虚拟机总是开始于一个 main()方法，这个方法必须是公有 public、返回 void、直接接收一个字符串数组。在程序执行时，你必须给 Java 虚拟机指明这个包含有 main()方法的类名。

Main()方法是程序的起点，它被执行的线程初始化为程序的初始线程。程序中其它的线程都由他来启动。Java 中的线程分为两种：守护线程 （daemon）和普通线程（non-daemon）。守护线程是 Java 虚拟机自己使用的线程，比如负责垃圾收集的线程就是一个守护线程。当然，你也可以把自己的程序设置为守护线程。包含 Main()方法的

初始线程不是守护线程。

只要 Java 虚拟机中还有普通的线程在执行，Java 虚拟机就不会停止。如果有足够的权限，你可以调用 exit()方法终止程序。

107、main 方法是做什么用的？

main 方法是 Java 程序的入口方法，JVM 在运行的时候会首先查找 main 方法。

108、不用 main 方法如何运行一个类？

不行，没有 main 方法我们不能运行 Java 类。
在 Java 7 之前，你可以通过使用静态初始化运行 Java 类。但是，从 Java 7 开始就行不通了。

109、main 方法如何传递参数？传递参数的类型是什么？能不能改变该参数类型？

109、String 数组，不能改变。

110、main 方法为什么是静态的？能不能改为非静态？

main()方法一定是静态的，如果 main()是非静态的那么在调用 main 方法时 JVM 就得实例化它的类。
不能改为非静态，main()方法必须声明为静态的，这样 JVM 才可以调用 main()方法而无需实例化它的类。
如果从 main()方法去掉“static”这个声明，虽然编译依然可以成功，但在运行时会导致程序失败。
在实例化时，还得调用类的构造函数。如果这个类的构造函数有参数，那么届时就会出现歧义。

112、main 方法能被重载吗？

可以，我们可以重载 main()方法。一个 Java 类可以有任意数量的 main()方法。

113、main 方法能被覆盖吗？

在 Java 中静态方法在编译时会编译在一起，main 方法是静态方法，所以你在 Java 中不能覆盖静态方法。

114、main 方法的返回类型是什么？能不能改变？

void，不能改变。Main 函数是一个特殊的函数可以被 jvm 虚拟机所识别

115、main 方法的作用域用什么修饰？能不能改变？

public，不能改变。

116、main 方法可以同步吗？

main 方法可以在 Java 中同步，synchronized 修饰符允许用于 main 方法的声明中，这样就可以在 Java 中同步 main 方法了。

117、main 方法可以终结吗？

可以在 Java 中终结 main 方法。

118、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)

答：接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承实体类，但前提是实体类必须有明确的构造函数

119、static 变量是什么含义？

static 是静态变量,就是变量值不随函数执行结束而消失，下次调用同一函数时，上次所赋予的值仍存在。