## 1. 什么叫做类的装载？

所谓装载就是寻找一个类或是一个接口的二进制形式并用该二进制形式来构造代表这个类或是这个接口的 class 对象的过程。在 Java 中，类装载器把一个类装入 Java 虚拟机中，要经过三个步骤来完成：装载、链接和初始化。

## 2. 类或者接口的信息一定是定义在 Class 文件中吗？

任何一个 Class 文件都对应着唯一一个类或接口的定义信息，但反过来说，类或接口并不一定都得定义在文件里（譬如类或接口也可以通过类加载器直接生成）。

## 3. 抽象的简述一下 Class 文件的结构。

Class 文件是一组以 8 位字节为基础单位的二进制流，各个数据项目严格按照顺序紧凑地排列在 Class 文件之中，中间没有添加任何分隔符，这使得整个 Class 文件中存储的内容几乎全部是程序运行的必要数据，没有空隙存在。当遇到需要占用 8 位字节以上空间的数据项时，则会按照高位在前的方式分割成若干个 8 位字节进行存储。

## 4. 无符号数和表是什么？

无符号数属于基本的数据类型，以 u1、u2、u4、u8 来分别代表 1 个字节、2 个字节、4 个字节和 8 个字节的无符号数，无符号数可以用来描述数字、索引引用、数量值或者按照 UTF-8 编码构成字符串值。
表是由多个无符号数或者其他表作为数据项构成的复合数据类型，所有表都习惯性地以“_info”结尾。表用于描述有层次关系的复合结构的数据，整个 Class 文件本质上就是一张表。

## 5. 说说 Class 文件的头四个字节。

每个 Class 文件的头 4 个字节称为魔数（Magic Number），它的唯一作用是确定这个文件是否为一个能被虚拟机接受的 Class 文件。

## 6. 说说字面量和符号引用是什么？

字面量比较接近于 Java 语言层面的常量概念，如文本字符串、声明为 final 的常量值等。符号引用则属于编译原理方面的概念，包括了下面三类常量：类和接口的全限定名（Fully Qualified Name）、字段的名称和描述符（Descriptor）、方法的名称和描述符。

## 7. 被 JVM 加载后的 Class 文件信息被存放在 JVM 的哪个区域中？

Minor Version、Major Version

## 8. 为什么 JAVA 字段、方法的符号引用不能在编译后就得到真正的内存入口地址，而 C/C++确可以？

Java 代码在进行 Javac 编译的时候，并不像 C 和 C++那样有“连接”这一步骤，而是在虚拟机加载 Class 文件的时候进行动态连接。也就是说，在 Class 文件中不会保存各个方法、字段的最终内存布局信息，因此这些字段、方法的符号引用不经过运行期转换的话无法得到真正的内存入口地址，也就无法直接被虚拟机使用。当虚拟机运行时，需要从常量池获得对应的符号引用，再在类创建时或运行时解析、翻译到具体的内存地址之中。

## 9. C/C++的编译时连接指的是什么？JAVA 的运行时动态连接又是指什么？

Java 代码在进行 Javac 编译的时候，并不像 C 和 C++那样有“连接”这一步骤，而是在虚拟机加载 Class 文件的时候进行动态连接。也就是说，在 Class 文件中不会保存各个方法、字段的最终内存布局信息，因此这些字段、方法的符号引用不经过运行期转换的话无法得到真正的内存入口地址，也就无法直接被虚拟机使用。当虚拟机运行时，需要从常量池获得对应的符号引用，再在类创建时或运行时解析、翻译到具体的内存地址之中。

## 10. UTF-8 缩略编码与普通 UTF-8 编码的区别是？

UTF-8 缩略编码与普通 UTF-8 编码的区别是：从'\u0001'到'\u007f'之间的字符（相当于 1～127 的 ASCII 码）的缩略编码使用一个字节表示，从'\u0080'到'\u07ff'之间的所有字符的缩略编码用两个字节表示，从'\u0800'到'\uffff'之间的所有字符的缩略编码就按照普通 UTF-8 编码规则使用三个字节表示。

## 11. 请叙述 JVM 是如何使用常量池的？

Class 文件中还有很多数据项都要引用常量池中的常量，字段表（field_info）、方法表（method_info）、属性表（attribute_info）都引用了。

### 12. Class 文件中由哪些数据来确定这个类的继承关系？

类索引（this_class）和父类索引（super_class）都是一个 u2 类型的数据，而接口索引集合（interfaces）是一组 u2 类型的数据的集合，Class 文件中由这三项数据来确定这个类的继承关系。

## 13. 在 Java 中描述一个字段可以包含什么信息？

字段（field）包括类级变量以及实例级变量，但不包括在方法内部声明的局部变量。可以包括的信息有：字段的作用域（public、private、protected 修饰符）、是实例变量还是类变量（static 修饰符）、可变性（final）、并发可见性（volatile 修饰符，是否强制从主内存读写）、可否被序列化（transient 修饰符）、字段数据类型（基本类型、对象、数组）、字段名称。

## 14. 要解释一下“简单名称”、“描述符”以及前面出现过多次的“全限定名”这三种特殊字符串的概念。

全限定名和简单名称很好理解，以代码清单 6-1 中的代码为例，“org/fenixsoft/clazz/TestClass”是这个类的全限定名，仅仅是把类全名中的“.”替换成了“/”而已，为了使连续的多个全限定名

之间不产生混淆，在使用时最后一般会加入一个“；”表示全限定名结束。
简单名称是指没有类型和参数修饰的方法或者字段名称，这个类中的 inc（）方法和 m 字段的简单名称分别是“inc”和“m”。
描述符的作用是用来描述字段的数据类型、方法的参数列表（包括数量、类型以及顺序）和返回值。

## 15. 描述符来描述方法 int indexOf（char[]source,int sourceOffset,int sourceCount,char[]target,int targetOffset,int targetCount,int fromIndex）

（[CII[CIII）I

## 16. 方法里的 Java 代码存在在 CLASS 文件的哪个区域？

Java 程序方法体中的代码经过 Javac 编译器处理后，最终变为字节码指令存储在 Code 属性内。

## 17. ＜init＞和＜clinit＞是什么方法？有什么作用？

类构造器“ < clinit > ”方法和实例构造器“ < init > ”方法。

## 18. 说说 transient 与 strictfp 关键字的作用？

transient：可否被序列化。
strictfp 关键字可应用于类、接口或方法。使用 strictfp 关键字声明一个方法时，该方法中所有的 float 和 double 表达式都严格遵守 FP-strict 的限制,符合 IEEE-754 规范。

## 19. 方法被编译后的执行指令被记录在 Class 文件的什么区域？

Code 属性。

## 20. 方法被编译后的栈深度和局部变量表所需的长度被记录在 Class 文件的什么区域？

Code 属性中的 max_stack 代表了操作数栈（Operand Stacks）深度的最大值。
Code 属性中的 max_locals 代表了局部变量表所需的存储空间。

## 21. 每个字节码指令占用几个字节的空间？JVM 有多少个字节码指令？

每个指令就是一个 u1 类型的单字节。
Java 虚拟机规范已经定义了其中约 200 条编码值对应的指令含义。

## 22. 请叙述 JVM 是如何利用 CODE 属性表来完成执行一个方法的？

## 23. 我们常用的 this 关键字在方法执行过程中被放在什么地方？

局部变量表。

## 24. 叙述栈帧的数据结构分别有哪几部分？

## 25. 简述 LineNumberTable 属性的作用？

LineNumberTable 属性用于描述 Java 源码行号与字节码行号（字节码的偏移量）之间的对应关系。它并不是运行时必需的属性，但默认会生成到 Class 文件之中，可以在 Javac 中分别使用-g：none 或-g：lines 选项来取消或要求生成这项信息。如果选择不生成 LineNumberTable 属性，对程序运行产生的最主要的影响就是当抛出异常时，堆栈中将不会显示出错的行号，并且在调试程序的时候，也无法按照源码行来设置断点。

## 26. 什么类型的表后面会跟随一个属性表？

字段表、方法表。

## 27. 简述一个属性表集合的每一个属性？

**Code 属性：**Java 程序方法体中的代码经过 Javac 编译器处理后，最终变为字节码指令存储在 Code 属性内。
**Exceptions 属性：**是列举出方法中可能抛出的受查异常（Checked Excepitons），也就是方法描述时在 throws 关键字后面列举的异常。
**LineNumberTable 属性：**用于描述 Java 源码行号与字节码行号（字节码的偏移量）之间的对应关系。
**LocalVariableTable 属性：**用于描述栈帧中局部变量表中的变量与 Java 源码中定义的变量之间的关系。
**SourceFile 属性：**用于记录生成这个 Class 文件的源码文件名称。
**ConstantValue 属性：**是通知虚拟机自动为静态变量赋值。
**InnerClasses 属性：**用于记录内部类与宿主类之间的关联。
**Deprecated 及 Synthetic 属性：**Deprecated 属性用于表示某个类、字段或者方法，已经被程序作者定为不再推荐使用，它可以通过在代码中使用@deprecated 注释进行设置。synthetic 属性代表此字段或者方法并不是由 Java 源码直接产生的，而是由编译器自行添加的。
**StackMapTable 属性：**在虚拟机类加载的字节码验证阶段被新类型检查验证器（Type Checker）使用（见 7.3.2 节），目的在于代替以前比较消耗性能的基于数据流分析的类型推导验证器。
**Signature 属性：**在 JDK 1.5 发布后增加到了 Class 文件规范之中，它是一个可选的定长属性，可以出现于类、属性表和方法表结构的属性表中。
**BootstrapMethods 属性：**在 JDK 1.7 发布后增加到了 Class 文件规范之中，它是一个复杂的变长属性，位于类文件的属性表中。这个属性用于保存 invokedynamic 指令引用的引导方法

限定符。

## 27. JAVA 的泛型实现为什么被业界称为假泛型？Java 的反射 API 为什么能获取泛型类型？

Java 语言的泛型采用的是擦除法实现的伪泛型，在字节码（Code 属性）中，泛型信息编译（类型变量、参数化类型）之后都通通被擦除掉。使用擦除法的好处是实现简单（主要修改 Javac 编译器，虚拟机内部只做了很少的改动）、非常容易实现 Backport，运行期也能够节省一些类型所占的内存空间。但坏处是运行期就无法像 C#等有真泛型支持的语言那样，将泛型类型与用户定义的普通类型同等对待，例如运行期做反射时无法获得到泛型信息。Signature 属性就是为了弥补这个缺陷而增设的，现在 Java 的反射 API 能够获取泛型类型，最终的数据来源也就是这个属性。

## 28. 代码实现提取 65536 的第二、第三个字节使用二进制格式打印出来。

## 29. 代码实现提取 65536 的第 13 到 16 位和第 5 到第 12 为前后、后前相接组成两个新的 int 值。

## 30.（byte1＜＜8）| byte2 的作用？

由于限制了 Java 虚拟机操作码的长度为一个字节（即 0～255），这意味着指令集的操作码总数不可能超过 256 条；又由于 Class 文件格式放弃了编译后代码的操作数长度对齐，这就意味着虚拟机处理那些超过一个字节数据的时候，不得不在运行时从字节中重建出具体数据的结构，如果要将一个 16 位长度的无符号整数使用两个无符号字节存储起来。

## 31. Java 虚拟机的解释器模型。

## 32. 说说这些指令执行的是什么操作：i2c、multianewarray、putfield、bastore、ladd、ineg、pop2、dup_x2、swap、if_icmpge、lookupswitch、jsr_w、invokevirtual 、 invokeinterface 、 invokespecial 、 invokestatic 、 invokedynamic。

i2c：
multianewarray：创建数组。
putfield：访问实例字段。
bastore：将一个操作数栈的值存储到数组元素中的指令。

ladd：加法指令。
ineg：取反指令。
pop2：将操作数栈的栈顶两个元素出栈。
dup_x2：复制栈顶两个数值并将双份的复制值重新压入栈顶。
swap：将栈最顶端的两个数值互换。
if_icmpge：条件分支。
lookupswitch：复合条件分支。
jsr_w：无条件分支。
invokevirtual：调用对象的实例方法。
invokeinterface：调用接口方法。
invokespecial：调用一些需要特殊处理的实例方法。
invokestatic：调用类方法（static 方法）。
invokedynamic：用于在运行时动态解析出调用点限定符所引用的方法，并执行该方法。

## 33. 为什么 Java 虚拟机提供的 int 类型的条件分支指令是最为丰富和强大的？

与前面算术运算时的规则一致，对于 boolean 类型、byte 类型、char 类型和 short 类型的条件分支比较操作，都是使用 int 类型的比较指令来完成，而对于 long 类型、float 类型和 double 类型的条件分支比较操作，则会先执行相应类型的比较运算指令（dcmpg、dcmpl、fcmpg、fcmpl、lcmp，见 6.4.3 节），运算指令会返回一个整型值到操作数栈中，随后再执行 int 类型的条件分支比较操作来完成整个分支跳转。由于各种类型的比较最终都会转化为 int 类型的比较操作，int 类型比较是否方便完善就显得尤为重要，所以 Java 虚拟机提供的 int 类型的条件分支指令是最为丰富和强大的。

## 34. 用管程（Monitor）是什么？有 monitorenter 和 monitorexit 是什么？

使用管程来支持方法级的同步。
同步一段指令集序列通常是由 Java 语言中的 synchronized 语句块来表示的，Java 虚拟机的指令集中有 monitorenter 和 monitorexit 两条指令来支持 synchronized 关键字的语义，正确实现 synchronized 关键字需要 Javac 编译器与 Java 虚拟机两者共同协作支持。