1：定义字符串方式：String str = “a”； 定义字符的方式：char str = 'a'；

2：接口中的方法默认都是public访问权限，所以一个类实现此接口时，在重写这个方法时只能是

public权限(子类重写父类方法，其访问权限不能低于父类中该方法的访问权限)

3：运行时异常又称为非检查型异常，一般出现在程序运行过程中（如数组越界，除数为0，空指针等）。

这种异常在程序中可不做处理，由java虚拟机自动处理。而非运行时异常又称为检查型异常

（如文件未找到异常等），这种异常需要在程序中手动处理，用try-catch捕获或者throws抛出。

4：以下关于ClassLoader的说法正确的是：ACD

A：ClassLoader就是用来动态加载class文件到内存当中用的

B：JVM在判定两个class是否相同时，只用判断类名相同即可，和类加载器无关

C：ClassLoader使用的是双亲委托模型来搜索类的

D：Java默认提供的三个ClassLoader是Boostrap ClassLoader，Extension ClassLoader，App ClassLoader

解析： JDK中提供了三个ClassLoader，根据层级从高到低为：

Bootstrap ClassLoader，主要加载JVM自身工作需要的类。

Extension ClassLoader，主要加载%JAVA\_HOME%\lib\ext目录下的库类。

Application ClassLoader，主要加载Classpath指定的库类，一般情况下这是程序默认类加载器，也是ClassLoader.getSystemClassLoader() 的返回值。（这里的Classpath默认指的是环境变量中配置的Classpath，但是可以在执行Java命令的时候使用-cp 参数来修改当前程序使用的Classpath）

JVM加载类的实现方式，我们称为 双亲委托模型：

如果一个类加载器收到了类加载的请求，他首先不会自己去尝试加载这个类，而是把这个请求委托给自己的父加载器，每一层的类加载器都是如此，因此所有的类加载请求最终都应该传送到顶层的Bootstrap ClassLoader中，只有当父加载器反馈自己无法完成加载请求时，子加载器才会尝试自己加载。

双亲委托模型的重要用途是为了解决类载入过程中的安全性问题：

假设有一个开发者自己编写了一个名为Java.lang.Object的类，想借此欺骗JVM。现在他要使用自定义ClassLoader来加载自己编写的java.lang.Object类。然而幸运的是，双亲委托模型不会让他成功。因为JVM会优先在Bootstrap ClassLoader的路径下找到java.lang.Object类，并载入它。

5： **static** String str0="0123456789";

**static** String str1="0123456789";

String str2=str1.substring(5);

String str3=**new** String(str2);

String str4=**new** String(str3.toCharArray());

str0=**null**;

假定str0,...,str4后序代码都是只读引用。Java 7中，以上述代码为基础，在发生过一次FullGC后，上述代码在Heap空间（不包括PermGen）保留的字符数为（15）

解析：这是一个关于java的垃圾回收机制的题目。垃圾回收主要针对的是堆区的回收，因为栈区的内存是随着线程而释放的。堆区分为三个区：年轻代（Young Generation）、年老代（Old Generation）、永久代（Permanent Generation，也就是方法区）。

 年轻代：对象被创建时（new）的对象通常被放在Young（除了一些占据内存比较大的对象）,经过一定的Minor GC（针对年轻代的内存回收）还活着的对象会被移动到年老代（一些具体的移动细节省略）。

年老代：就是上述年轻代移动过来的和一些比较大的对象。Minor GC(FullGC)是针对年老代的回收

永久代：存储的是final常量，static变量，常量池。

str3,str4都是直接new的对象，而substring的源代码其实也是new一个string对象返回。

经过fullgc之后，年老区的内存回收，则年轻区的占了15个，不算PermGen。所以答案为15（5+5+5）

6：如果希望监听TCP端口9000，应怎样创建socket？ B

A：new Socket("localhost",9000);

B：new ServerSocket(9000);

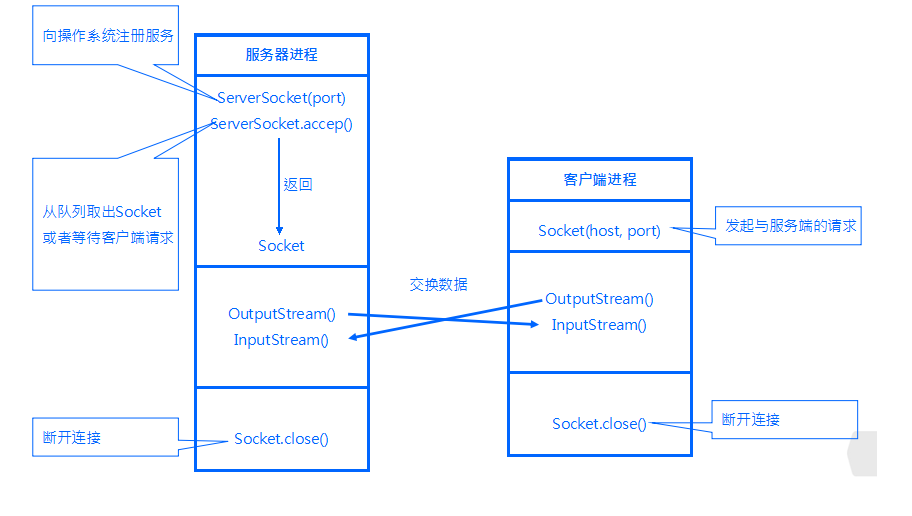
C：new Socket(9000);

D：new ServerSocket("localhost",9000);

解析：ServerSocket(int port) 是服务端绑定port端口，调accept()监听等待客户端连接，它返回一个连接队列中的一个socket。

Socket(InetAddress address , int port)是创建客户端连接主机的socket流，其中InetAddress是用来记录主机的类，port指定端口。

socket和servletSocket的交互如下图所示：



7：关于静态初始化块说法正确的是：ABC

A：无法直接调用静态初始化块

B：在创建第一个实例前或引用任何静态成员之前，将自动调用静态初始化块来初始化

C：静态初始化块既没有访问修饰符，也没有参数

D：在程序中，用户可以控制何时执行静态初始化块

解析：静态初始化块在类加载的时候就会执行，且只执行一次，无法直接调用。初始化块在new一个该类对象时执行，且是在构造方法前执行。静态初始化块，初始化块，构造函数的执行顺序如下： 静态初始化块 > 初始化块> 构造函数

8：java8中，忽略内部接口的情况，不能用来修饰interface里的方法的有（AC）

A：private

B：public

C：protected

D：static

解析：Java8的接口方法可以有如下定义

only public, abstract, default, static and strictfp are permitted

9：下列关于管道（Pipe）通信的叙述中，正确的是（A）？

A：进程对管道进行读操作和写操作都可能被阻塞

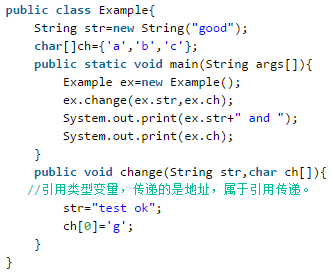
B：一个管道只能有一个进程或一个写进程对其操作

C：一个管道可实现双向数据传输

D：管道的容量仅受磁盘容量大小限制

解析：A.正确，因为管道为空，读操作会被阻塞；管道满了，写操作会被阻塞  
B.可以有多个进程对其读；也可以有多个进程写，只不过不能同时写。并且题目 没有说“同时”，B不对  
C.匿名管道只能单向；命名管道可以双向；所以C过于绝对  
D.管道是内存中的，所以D不对

10：以下程序的运行结果是：good and gbc



解析：在[Java](http://lib.csdn.net/base/java" \t "https://www.nowcoder.com/test/question/_blank)中，每一个new语句返回的都是一个指针的引用，只不过在大多数情况下Java不关心如何操作这个指针，****唯一需要注意的是在给函数传递对象的时候****传值和引用中有静态变量，私有变量，clone等问题，对于基本数据类型和对象变量，存在不一样的操作：

基本类型变量：传递****值****的副本（副本变，本身不变）

对象变量：传递****引用****的副本（副本变，本身也变）【即复制指向地址的指针】

****不论Java参数的类型是什么，一律传递的都是参数的副本****

在Java中存在一个比较特殊的类型，String，也是对象型变量，只不过String是一个非可变类，使得值传递和引用传递没有区别。

11：JAVA语言的下面几种数组复制方法中，哪个效率最高？ B

A：for循环逐一复制

B：System.arraycopy

C：Arrays.copyof

D：使用clone方法

解析：System.arraycopy()源码。可以看到是native方法：native关键字说明其修饰的方法是一****个原生态方法****，方法对应的实现不是在当前文件，而是在用其他语言（****如C和C++****）实现的文件中。 可以将native方法比作Java程序同Ｃ程序的接口。其效率是最高的。clone方法效率最差。

12：以下哪些jvm的垃圾回收方式采用的是复制算法回收？ （AD）

A：新生代串行收集器

B：老年代串行收集器

C：并行收集器

D：新生代并行回收收集器

E：老年代并行回收收集器

F：cms收集器

解析：两个最基本的java回收算法：复制算法和标记清理算法

复制算法：两个区域A和B，初始对象在A，继续存活的对象被转移到B。此为新生代最常用的算法

标记清理：一块区域，标记要回收的对象，然后回收，一定会出现碎片，那么引出

标记-整理算法：多了碎片整理，整理出更大的内存放更大的对象

两个概念：新生代和年老代

新生代：初始对象，生命周期短的

永久代：长时间存在的对象

整个java的垃圾回收是新生代和年老代的协作，这种叫做分代回收。

P.S：**Serial New收集器是针对新生代的收集器，采用的是复制算法**

**Parallel New（并行）收集器，新生代采用复制算法，老年代采用标记整理**

**Parallel** **Scavenge（并行）收集器，针对新生代，采用复制收集算法**

**Serial Old（串行）收集器，新生代采用复制，老年代采用标记整理**

**Parallel** **Old（并行）收集器，针对老年代，标记整理**

**CMS收集器，基于标记清理**

**G1收集器：整体上是基于标记** **整理** **，局部采用复制**

**综上：新生代基本采用复制算法，老年代采用标记整理算法。cms采用标记清理。**