**1、**重载和重写的区别?构造器 Constructor 是否可被 override?

重载： 发生在同一个类中，方法名必须相同，参数类型不同、个数不同、顺序不同，方法返回值和访问修饰符可以不同，发生在编译时。

•重写： 发生在父子类中，方法名、参数列表必须相同，返回值范围小于等于父类，抛出的异常范围小于等于父类，访问修饰符范围大于等于父类；

继承的时候，父类的私有属性和构造方法并不能被继承，所以 Constructor 也就不能被 override（重写）,但是可以 overload（重载）,一个类中有多个构造函数的情况。

**2、**继承、多态

继承：子类拥有父类对象所有的属性和方法（包括私有属性和私有方法），但是父类中的私有属性和方法子类是无法访问，只是拥有。

**3.**多态

在Java中有两种形式可以实现多态：继承（多个子类对同一方法的重写）和接口（实现接口并覆盖接口中同一方法）。

**4.** String str="abcd"与 String str1=new String("acbd")一样吗？str和str1相等吗?

第一种方式先检查字符串常量池中有没有"abcd"，如果字符串常量池中没有，则创建一个，然后str指向字符串常量池中的对象，如果有，则直接将str指向字符串常量池中的"abcd"；

第二种方式是直接在堆内存空间创建一个新的对象。

可能创建一个对象也可能创建两个对象。

**5.** String StringBuffer 和 StringBuilder 的区别是什么?String 为什么是不可变的?

（1）String 类中使用 final 关键字修饰字符数组来保存字符串，private　final　char　value[]，是不可变的。

（2）StringBuilder 与 StringBuffer 都继承自 AbstractStringBuilder 类，在 AbstractStringBuilder 中也是使用字符数组保存字符串char[]value ,没有用 final 关键字修饰，是可变的。

（3）操作少量的数据: 适用String

（4）单线程操作字符串缓冲区下操作大量数据: 适用StringBuilder

（5）多线程操作字符串缓冲区下操作大量数据: 适用StringBuffer

**6.** 接口和抽象类的区别是什么？抽象类必须要有抽象方法吗？抽象类能使用 final 修饰吗？

（1）接口的方法默认是 public，而抽象类可以有非抽象的方法，抽象类不能用final来修饰。

（2）一个类可以实现多个接口，但最多只能实现一个抽象类。

（3）一个类实现接口的话要实现接口的所有方法，而抽象类不一定。

（4）接口不能用 new 实例化，但可以声明，但是必须引用一个实现该接口的对象。

（5）抽象类是对类的抽象，是一种模板设计，而接口是对行为的抽象。

（6）抽象类中不一定包含抽象方法，但是包含抽象方法的类一定要被声明为抽象类。

**7.** == 与 equals(重要)

== :判断两个对象的地址是不是相等。判断两个对象是不是同一个对象(基本数据类型==比较的是值，引用数据类型==比较的是内存地址)。

equals() :判断两个对象是否相等。有两种使用情况：

（1）没有覆盖 equals() 方法，等价于通过“==”比较这两个对象。

（2）覆盖了 equals() 方法。若它们的内容相等，则返回 true (即，认为这两个对象相等)。

**8.** hashCode 与 equals (重要)

（1）如果两个对象相等，则hashcode一定也是相同的;

（2）两个对象相等,对两个对象分别调用equals方法都返回true;

（3）两个对象有相同的hashcode值，它们也不一定是相等的。

（4）equals 方法被覆盖过，则 hashCode 方法也必须被覆盖

（5）hashCode() 的默认行为是对堆上的对象产生独特值。如果没有重写 hashCode()，则该 class 的两个对象无论如何都不会相等（即使这两个对象指向相同的数据）

**9.**反射

反射机制优缺点

优点： 运行期类型的判断，动态加载类，提高代码灵活度。

•缺点：反射相当于一系列解释操作，通知 JVM 要做的事情，性能比直接的java代码要慢很多。

反射是框架设计的灵魂。

**10.** Java 中 IO 流分为几种？BIO,NIO,AIO 有什么区别？

(1)分类

按照流的流向分，可以分为输入流和输出流；

按照操作单元划分，可以划分为字节流和字符流；

按照流的角色划分为节点流和处理流。

Java I0流的40多个类都是从如下4个抽象类基类中派生出来的。

•InputStream/Reader: 所有的输入流的基类，前者是字节输入流，后者是字符输入流。

•OutputStream/Writer: 所有输出流的基类，前者是字节输出流，后者是字符输出流。

(2) BIO,NIO,AIO 有什么区别？

BIO (Blocking I/O): 同步阻塞I/O模式，数据的读取写入必须阻塞在一个线程内等待其完成。不适合处理高并发的场景。

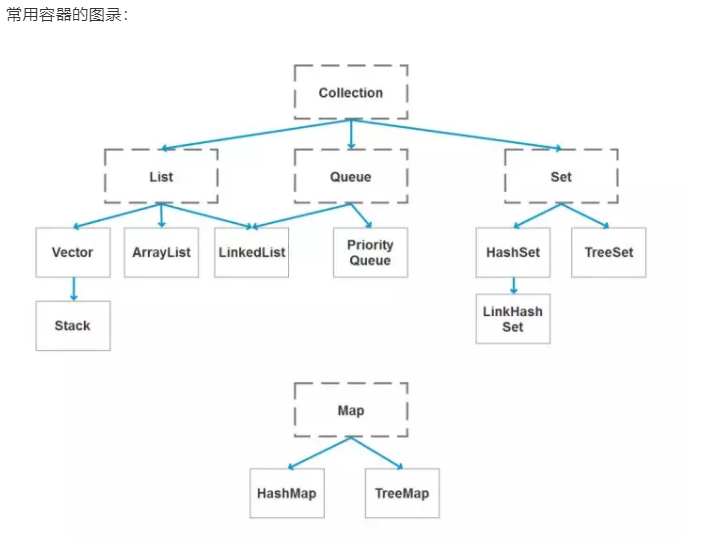
•NIO (New I/O): NIO是一种同步非阻塞的I/O模型，NIO中的N可以理解为Non-blocking，不单纯是New。它支持面向缓冲的，基于通道的I/O操作方法。对于高负载、高并发的（网络）应用，应使用 NIO 的非阻塞模式来开发。

•AIO (Asynchronous I/O): AIO 也就是 NIO 2。在 Java 7 中引入了 NIO 的改进版 NIO 2,它是异步非阻塞的IO模型。异步 IO 是基于事件和回调机制实现的，也就是应用操作之后会直接返回，不会堵塞在那里，当后台处理完成，操作系统会通知相应的线程进行后续的操作。

**11.**ArrayList 和 LinkedList 的区别是什么？

最明显的区别是 ArrrayList底层的数据结构是数组，支持随机访问，而 LinkedList 的底层数据结构是双向循环链表，不支持随机访问。

**12**.JAVA常见容器有哪些？



**13.** Collection 和 Collections 有什么区别？

java.util.Collection 是一个集合接口（集合类的一个顶级接口）。它提供了对集合对象进行基本操作的通用接口方法。Collection接口在Java 类库中有很多具体的实现。

(1)Collection接口的意义是为各种具体的集合提供了最大化的统一操作方式，其直接继承接口有List与Set。

(2)Collections则是集合类的一个工具类/帮助类，其中提供了一系列静态方法，用于对集合中元素进行排序、搜索以及线程安全等各种操作。

**14**.https的原理

https = http + ssl

(1)对称加密

对称加密指的就是加密和解密使用同一个秘钥。

对称加密算法：DES。

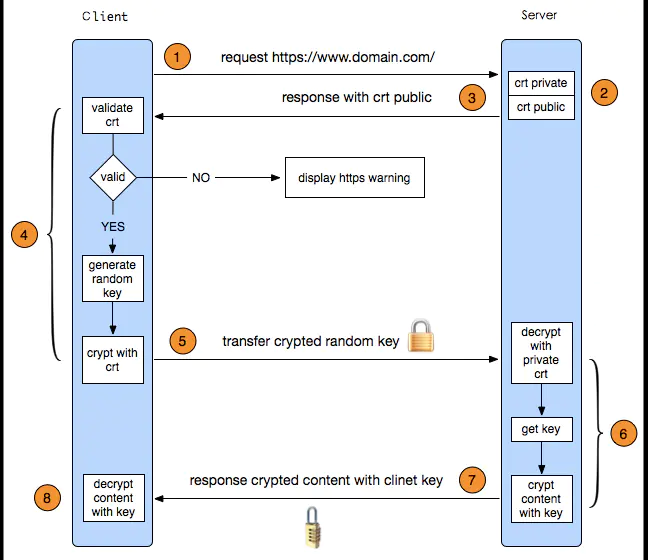
(2)非对称加密

加密和解密使用不同的秘钥，一把公钥，一把私钥。

常见的非对称加密算法：RSA。

(3)区别

对称加密算法加解密的效率要高。但是缺陷在于对于秘钥的管理上安全性较低。所以非对称加密算法加解密效率低，但是秘钥管理上比较安全。



HTTPS在传输的过程中会涉及到三个密钥:

服务器端的公钥和私钥，用来进行非对称加密

客户端生成的随机密钥，用来进行对称加密

一个HTTPS请求实际上包含了两次HTTP传输，可以细分为8步。

(1).客户端向服务器发起HTTPS请求，连接到服务器的443端口

(2).服务器端有一个密钥对，即公钥和私钥，是用来进行非对称加密使用的，服务器端保存着私钥，不能将其泄露，公钥可以发送给任何人。

(3).服务器将自己的公钥发送给客户端。

(4).客户端收到服务器端的证书之后，会对证书进行检查，验证其合法性，如果发现发现证书有问题，那么HTTPS传输就无法继续。严格的说，这里应该是验证服务器发送的数字证书的合法性，关于客户端如何验证数字证书的合法性。如果公钥合格，那么客户端会生成一个随机值，这个随机值就是用于进行对称加密的密钥，我们将该密钥称之为client key，即客户端密钥，这样在概念上和服务器端的密钥容易进行区分。***然后用服务器的公钥对客户端密钥进行非对称加密，这样客户端密钥就变成密文了，至此，HTTPS中的第一次HTTP请求结束。***

(5).客户端会发起HTTPS中的第二个HTTP请求，将加密之后的客户端密钥发送给服务器。

(6).***服务器接收到客户端发来的密文之后，会用自己的私钥对其进行非对称解密，解密之后的明文就是客户端密钥，然后用客户端密钥对数据进行对称加密，这样数据就变成了密文。***

(7).然后服务器将加密后的密文发送给客户端。

(8).客户端收到服务器发送来的密文，用客户端密钥对其进行对称解密，得到服务器发送的数据。这样HTTPS中的第二个HTTP请求结束，整个HTTPS传输完成。

**15.** 私钥、公钥、数字签名

(1)**公钥是公开的，私钥是私有的**，所以Alice给Bob发消息，就要使用Bob公开的公钥来做加密，Bob再用自己的私钥进行解密即可。**简单来说，公钥加密，私钥解密。**

**数字签名刚好相反**，比如大家想对Alice的签名进行验证，那么大家能获取的就是Alice公开的公钥，而签名则由Alice用自己的私钥进行签名。**简单来说，私钥签名，公钥验证。**

(2)**签名：通常的理解就是签字画押。**用来作为，某个人的同意证明。如何证明呢？B　有自己的私钥，别人没有。如果，用B的私钥对文件进行加密。把文件发给A，A再使用B的公钥，进行解密。如果解密成功，那么，就证明是用B的私钥加密的，从而证明了是B本人亲自操作的。

文档解释：https://blog.csdn.net/kswkly/article/details/83617944