**1. 怎么知道要跟哪个端口沟通？**

在Java中，要确定要与哪个端口进行通信，你需要使用Socket类。Socket类提供了与网络上的特定端口进行通信的能力。

首先，你需要创建一个Socket对象，并指定要连接的主机和端口号。例如，如果你想连接到主机名为"example.com"的服务器上的端口号为8080的端口，你可以这样做：



在上面的代码中，我们使用Socket类的构造函数来创建一个Socket对象，并传入主机名和端口号作为参数。如果连接成功，你就可以在Socket对象上执行进一步的通信操作。

请注意，如果连接失败，将抛出IOException异常。因此，你需要在try-catch块中捕获并处理该异常。

**2. 怎么知道要跟哪个端口沟通？**

在一般情况下，不同程序是不能共享同一个端口的。每个端口在特定时间只能由一个程序使用。每个端口都有一个唯一的标识符，用于在网络上识别和定位特定的应用程序或服务。当一个程序在一个端口上监听连接时，其他程序不能使用相同的端口。然而，有一些特殊情况下，多个程序可以共享同一个端口。这通常发生在使用多线程或多进程的服务器应用程序中。在这种情况下，服务器程序可以创建多个线程或进程来处理来自不同客户端的连接，每个连接都使用相同的端口。另外，还有一些特殊的协议和技术，如HTTP的反向代理、负载均衡器和网络地址转换（NAT），它们可以使多个程序共享同一个端口。总的来说，通常情况下，不同程序不能共享同一个端口，但在特定的情况下，可以使用一些技术来实现多个程序共享同一个端口。

**3. 数据是怎么通过光纤/光缆传输的？**

光纤是由非常纯净的玻璃或塑料制成的细长线材。它具有非常高的折射率，可以使光信号在其中传播。数据通过光纤传输时，实际上是通过光信号的变化来表示和传递的。

1. 发送端：首先，数据被转换为光信号。这通常是通过将电信号转换为光信号的光发射器（例如激光二极管）来实现的。光发射器会发出脉冲光信号，其中每个脉冲代表一个比特（0或1）。2. 传输：光信号通过光纤中的内核（纤芯）传播。内核是由高折射率的材料制成的，使光信号在其中反射和传输。内核被包裹在一个低折射率的外层（包层）中，以防止光信号泄漏。3. 接收端：在接收端，光信号到达光接收器。光接收器是一种光电二极管或光探测器，它将光信号转换回电信号。光接收器会检测光信号的强度和变化，并将其转换为相应的电信号。4. 数据处理：接收到的电信号经过进一步的处理和解码，以恢复原始的数据。

**4. 一个数字信号有8个电平。那么每个电平需要多少个位？**

Log28 = 3；每个信号电平用3位表示。

**5. 比特率是1秒钟发送的位数，单位是bps，或者b/s**

假定我们需要每分钟100页的速率下载文本文档。所需的通道比特率是多少？（假设一页平均包含24行，每行80个字符）

100×24×80×8/60 = 25600bps = 125.6kbps

**6.香农**

通道的容量=带宽🗙log2(1+SNR)

我们可以计算一条常规电话线路理论上的最高比特率。通常情况下，电话线路带宽为3KHz。信噪比通常为3162。对于这一通道，其容量计算为：

文本

描述已自动生成

文本, 信件

描述已自动生成

**7.输出流**

文字图案

中度可信度描述已自动生成 图示

描述已自动生成

使用write(byte[] data)或write(byte[] data, int offset, int length)通常比单个比特的传送效率要高。

文本, 信件

描述已自动生成

**8.输入流**

文本

描述已自动生成文本, 信件

描述已自动生成

**9.过滤器**

文本, 信件

描述已自动生成手机屏幕截图

描述已自动生成

**10.缓冲流**

手机屏幕截图

中度可信度描述已自动生成

文本

描述已自动生成

**11.Buffer和print流应用场景**

图片包含 图形用户界面

描述已自动生成 图片包含 图形用户界面

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成 图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**12.Buffer应用**

日程表

描述已自动生成

**13.文件字节流。**字符串12个字节最多

**图片包含 文本

描述已自动生成**

**14.DataInputStream类和DataOutputStream类。**一个整数值和一个字符串

文本

描述已自动生成

**15.字节数组流**

图形用户界面, 文本, 应用程序

描述已自动生成

**16.字符流数组**

**图片包含 文本

描述已自动生成**

**问题：将data.txt文件中保存的信息缓冲输入到程序中，编程时：如何创建读入流？**

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

**数据格式不一致，需编码转换，将JSP或Servlet中的请求数据从UTF8转换为GB2312，编程时：如果直接从http流中提取中文数据，提取的结果通常是“?????”，因此需要转换成可理解的中文字符，把UTF8转换成GB2312。如何编程实现转换？**

文本

描述已自动生成

**进程与线程**

**进程**：具有一定独立功能的程序关于某个数据集合上的一次运行活动，进程是系统**进行资源分配和调度**的一个**独立单元。**

* **线程**：

1 进程的一个实体。

2 是CPU调度和分派的基本单位。它是比进程更小的能独立运行的基本单位。

3 线程自己基本上不拥有系统资源，只拥有一点在运行中必不可少的资源（程序计数器，一组寄存器和栈。）

4 可与同属一个进程的其他的线程共享进程所拥有的全部资源。

* 一个线程可以创建和撤销另一个线程，同一个进程中的多个线程可以并发执行。
* 一个程序至少有一个进程，一个进程**至少有一个线程**（主线程）。
* 进程拥有**独立内存单元**，多个线程共享内存。
* 创建**进程开销大**，线程**开销较小**。

**提升并发量的方法**

* + 过去：以进程为单位
    - 多进程：支持几百个进程
    - 进程池：支持上千个进程
  + 当前：以线程为单位
    - 多线程：支持几千个线程
    - 线程池：支持上万个线程
* 其他
  + 异步I/O
  + 多个冗余服务器

图形用户界面, 文本, 电子邮件

描述已自动生成 图形用户界面, 文本

描述已自动生成

图示

描述已自动生成 图示

描述已自动生成

文本

描述已自动生成 文本

描述已自动生成

同步块

* 同一个对象上同步的代码串行执行
* 不同对象上同步的代码可并行

多个线程会使用LogFile对象的引用获得当前日期和时间，写入到日志文件中。如何编程处理，避免写入混乱？

文本

描述已自动生成

文本

描述已自动生成

java实现线程池有哪些类，给出它们之间的关系，和各自的类结构图

java.util.concurrent.Executor：是一个顶层接口，定义了执行任务的方法。

java.util.concurrent.ExecutorService：是Executor接口的子接口，扩展了一些管理和控制线程池的方法。

java.util.concurrent.ThreadPoolExecutor：是ExecutorService接口的实现类，是Java中最常用的线程池实现。它使用一个线程池来执行提交的任务。

java.util.concurrent.ScheduledExecutorService：是ExecutorService接口的子接口，提供了在给定的延迟后或定期执行任务的功能

图示

描述已自动生成

* **使用Executor线程池编程框架时，定义计算任务的类应实现（B）接口？**

A Future

B Callable

C Executor

D Runnable

**使用多线程编程查找一个很大的数字数组中的最大值。P72-73**

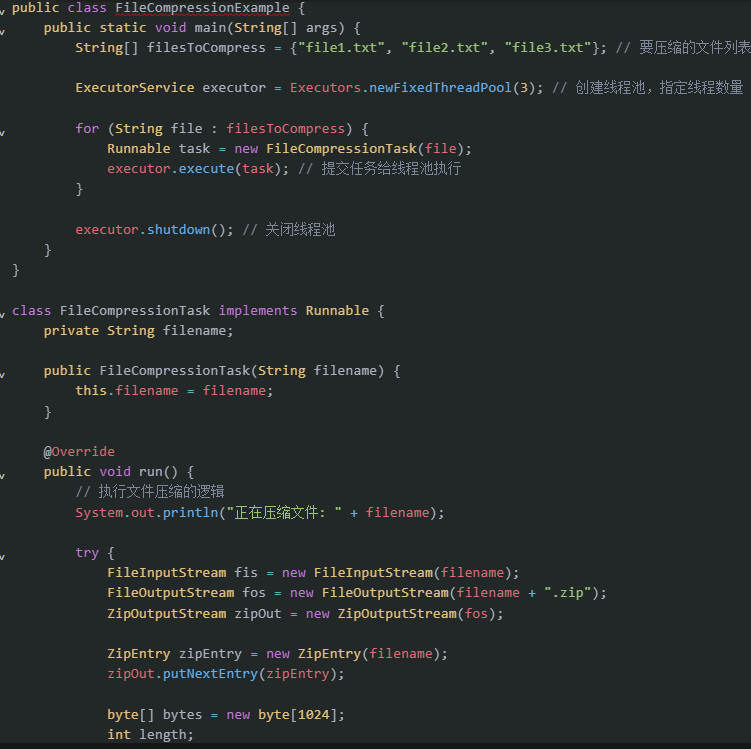
文本

描述已自动生成

文本

描述已自动生成

使用线程池实现压缩文件编程



文本

描述已自动生成

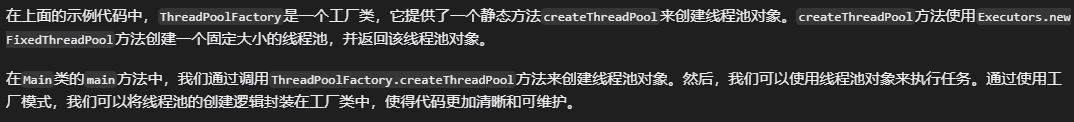
**Java实现线程池使用了工厂模式，什么是工厂模式？请试使用这种设计模式**

工厂模式是一种创建对象的设计模式，它提供了一种将对象的创建逻辑封装在一个独立的类中的方式。工厂模式通过定义一个公共的接口或抽象类来创建对象，而不是直接使用new关键字来实例化对象。这样可以将对象的创建与使用代码解耦，提高代码的可维护性和灵活性。

在Java中，可以使用工厂模式来创建线程池对象。下面是一个示例代码，展示了如何使用工厂模式创建线程池：

文本

描述已自动生成



**IPaddress**

* 由于InetAddress没有public的构造方法，因此，要想创建InetAddress对象，必须得依靠它的四个静态方法；
* 本机的InetAddress对象：通过getLocalHost方法得到；
* 远程主机的InetAddress对象：通过getByName、getAllByName和getByAddress得到；
* 使用**getLocalHost**可以得到描述**本机IP**的InetAddress对象。
* 方法定义：

public static InetAddress getLocalHost() throws

UnknownHostException

**当本机绑定了多个IP地址时候，只返回第一个地址！**

**`getLocalHost`**和**`getByName`**是Java中用于获取主机信息的两个方法，它们有以下区别：

**1. `getLocalHost`方法：**

- `getLocalHost`方法是`InetAddress`类的静态方法，用于获取本地主机的`InetAddress`对象。

- 它返回一个表示本地主机的`InetAddress`对象，可以通过该对象获取主机的IP地址和主机名。

- 通常情况下，`getLocalHost`方法会返回本地机器的IP地址和主机名，但在某些情况下可能返回一个虚拟的IP地址或主机名。

**2. `getByName`方法：**

- `getByName`方法也是`InetAddress`类的静态方法，用于根据主机名或IP地址获取对应的`InetAddress`对象。

- 它接受一个主机名或IP地址作为参数，并返回一个表示该主机或IP地址的`InetAddress`对象。

- 如果传入的是一个主机名，`getByName`方法会尝试解析该主机名并返回对应的`InetAddress`对象。如果解析失败，会抛出`UnknownHostException`异常。

* 当host的值是localhost时，返回的IP一般是127.0.0.1。

- 如果传入的是一个IP地址，`getByName`方法会直接返回对应的`InetAddress`对象。

如果host是IP地址，无论这个IP地址是否存在，getByName方法都会返回这个IP地址（因此getByName并不验证IP地址的正确性）。

**总结：**

- `getLocalHost`方法用于获取本地主机的`InetAddress`对象，返回本地机器的**IP地址**和**主机**

- `getByName`方法用于根据主机名或IP地址获取对应的`InetAddress`对象，**可以解析主机名并返回对应的IP地址。**

**getAllByName**

* 使用getAllByName方法可以从DNS上得到域名对应的所有的IP。这个方法返回一个InetAddress类型的数组。这个方法的定义如下：
* 与getByName方法一样，当host不存在时，getAllByName也会抛出UnknowHostException异常，getAllByName也不会验证IP地址是否存在。

图形用户界面, 文本, 应用程序

描述已自动生成 **图形用户界面, 文本

描述已自动生成**

**文本

描述已自动生成 文本

描述已自动生成**

**getByAddress**

**文本

描述已自动生成 文本

低可信度描述已自动生成**

**程序安全性问题**

**文本

描述已自动生成**

**DNS缓存**

**文本, 信件

描述已自动生成 文本

描述已自动生成**

根据给定的IP地址获取对应的规范主机名，并将其打印输出。

**图形用户界面, 文本

描述已自动生成**

**图形用户界面, 文本, 应用程序

描述已自动生成**

**使用getHostName方法获得域名**

* 总结：三种方式，只有通过使用IP地址创建的InetAddress对象调用getHostName方法时才访问DNS服务器。在其他情况，getHostName方法并不会访问DNS服务器，而是直接将域名或本机名返回。

**文本

描述已自动生成**

**文本

描述已自动生成**

* **说明使用域名创建的InetAddress对象在使用getHostName方法时并不访问DNS服务器。**
* **而使用IP地址创建的InetAddress对象在使用getHostName方法时需要访问DNS服务器。**

**使用getCanonicalHostName方法获得主机名**

**文本

描述已自动生成**

**图形用户界面, 文本

描述已自动生成**

**使用getAddress方法获得IP地址**

**文本

描述已自动生成 文本

描述已自动生成**

**文本

描述已自动生成**

* **如何快速判断垃圾邮件发送者？**

**文本

描述已自动生成**

**文本

描述已自动生成**

**文本

描述已自动生成**

**URL**

**文本, 信件

描述已自动生成**

**文本

描述已自动生成**

**试从程序输入输出处理数据的角度思考并解释一下，从PC浏览器地址栏输入一个url，到可以在浏览器中看到这个网页内容，什么程序从哪里获得了什么数据？处理了什么数据？**

**文本

描述已自动生成**

**文本

描述已自动生成**

**下载一个Web页面，思路：从命令行读取一个URL（http://www.oreilly.com），从这个URL打开一个InputStream，将此InputStream串链到默认编码方式的InputStreamReader，使用其read()方法从文件读取连续的字符，将字符在控制台显示输出。**

文本

描述已自动生成

**文本

描述已自动生成**

**在上面的代码中，我们添加了一个命令行参数<fileType>，用于指定要下载的文件类型。程序会根据文件类型判断是否下载文件。**

**图示

中度可信度描述已自动生成**

**文本

描述已自动生成**

**URI**

* + **编程实现网络数据获取功能时选用URL类**
  + **比如 想要下载一个url的内容。**
  + **编程用于解析和处理统一资源定位符相关的字符串时选用URI类。**
  + **URI类无网络获取功能。**
  + **比如 想要表示一个XML命名空间。**

**文本, 信件

描述已自动生成**

**一个360购物网站的表单处理程序如何处理搜索访问表单？http://gouwu.360.cn，利用url**

**文本

描述已自动生成**

**文本

描述已自动生成**

**如何编程实现网页加密，使某个或某些特定网页的查看受口令保护？**

**文本

描述已自动生成**

**文本

描述已自动生成**

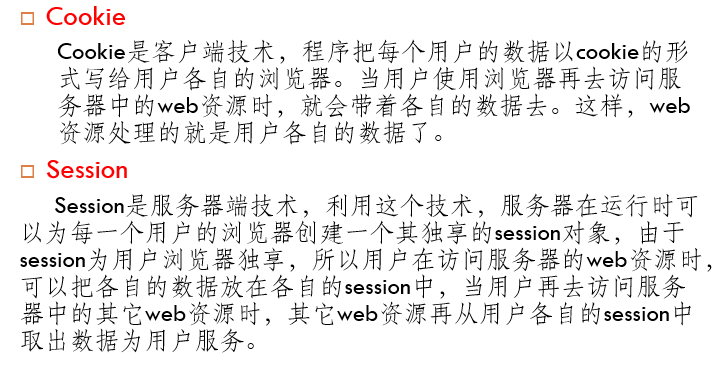
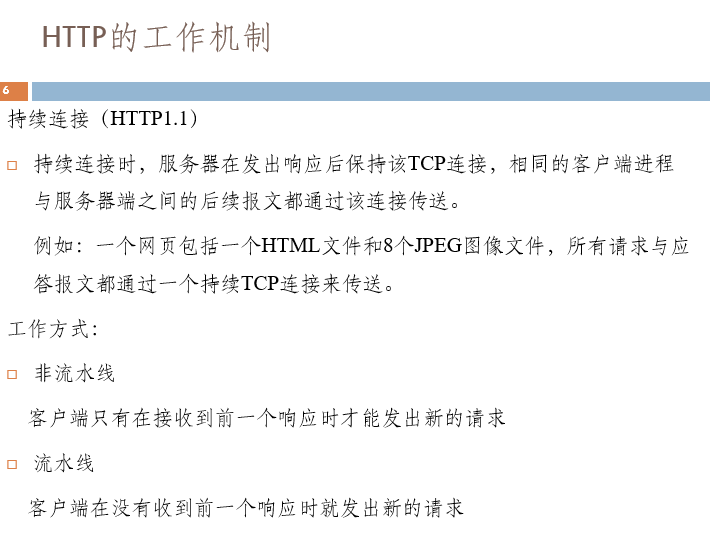
**文本

描述已自动生成**

**HTTP的工作机制**

**文本

描述已自动生成**

****

**客户端**

**文本

描述已自动生成**

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

服务器

文本

描述已自动生成

* 编程自定义一个简单的cookie策略类,阻塞来自.gov域的cookie,接受所有其他域的cookie.

文本

描述已自动生成

文本

描述已自动生成

**使用`URL`实现HTTP请求有以下几个好处：**

1. 简单易用：`URL`类提供了简单易用的API，使得发送HTTP请求变得简单和直观。你只需要创建一个`URL`对象，然后使用它的方法来打开连接、设置请求属性、发送请求和获取响应。

2. 内置支持：`URL`类内置了对HTTP协议的支持，可以直接处理HTTP请求和响应。你不需要编写复杂的网络通信代码，只需使用`URL`类提供的方法即可完成HTTP通信。

3. 标准库支持：`URL`类是Java标准库的一部分，无需额外的依赖。这意味着你可以在任何支持Java的平台上使用`URL`类，而无需引入其他第三方库。

4. 跨平台性：由于`URL`类是Java标准库的一部分，因此可以在不同的操作系统和平台上使用相同的代码。这使得你的HTTP请求代码具有跨平台性，可以在不同的环境中运行。

5. 可扩展性：`URL`类提供了基本的HTTP请求和响应功能，但如果需要更高级的功能，你可以使用`HttpURLConnection`类进行扩展。`HttpURLConnection`是`URL`类的子类，提供了更多的HTTP相关功能，如处理重定向、处理Cookies、设置请求方法等。

尽管`URL`类在处理简单的HTTP请求时非常方便，但对于复杂的HTTP通信，可能需要使用更专业的HTTP客户端库，如Apache HttpClient或OkHttp。这些库提供了更丰富的功能和更好的性能，适用于更复杂的HTTP场景。



文本

描述已自动生成

文本

描述已自动生成

URLConnection



文本

描述已自动生成

`URLConnection`是Java中用于建立和管理与URL之间的连接的抽象类。它是`java.net`包中的一部分，提供了许多方法来发送和接收HTTP请求和响应。

`URLConnection`类的主要功能包括：

1. 打开连接：使用`openConnection()`方法打开与URL的连接，返回一个`URLConnection`对象。可以通过`URL`对象的`openConnection()`方法来获取连接。

2. 设置请求属性：可以使用`setRequestProperty()`方法设置请求头的属性，如User-Agent、Accept、Content-Type等。

3. 发送请求：可以使用`getOutputStream()`方法获取输出流，并将请求体数据写入输出流。对于GET请求，可以使用`getInputStream()`方法获取输入流，获取服务器的响应数据。

4. 获取响应信息：可以使用`getResponseCode()`方法获取HTTP响应的状态码，使用`getHeaderField()`方法获取响应头的字段值，使用`getInputStream()`方法获取输入流，读取响应体的数据。

5. 断开连接：使用`disconnect()`方法关闭连接。

`URLConnection`是一个抽象类，具体的实现类是`HttpURLConnection`，用于处理HTTP连接。`HttpURLConnection`提供了更多的HTTP相关功能，如设置请求方法、处理重定向、处理Cookies等。

使用`URLConnection`可以实现与URL的连接、发送HTTP请求和接收响应。它提供了一种简单而灵活的方式来与Web服务器进行通信，并获取所需的数据。

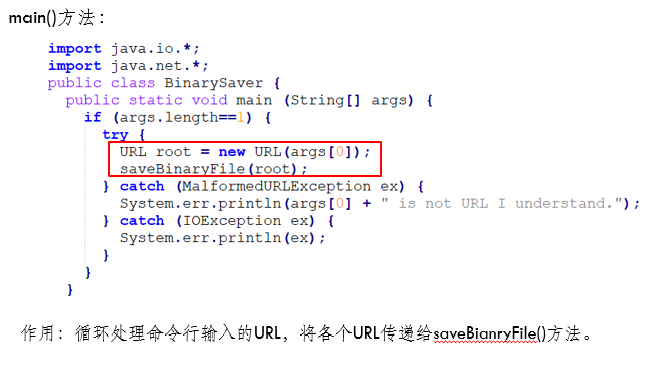
* URLConnection提供了对HTTP头信息访问
* URLConnection可以配置发给服务器的参数
* URLConnection可向服务器发送数据

文本

描述已自动生成

文本, 信件

描述已自动生成



图形用户界面, 应用程序

中度可信度描述已自动生成

文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

* **编程实现：提供任意一个url，在控制台显示输出HTTP服务器响应整个头部。**

**文本

描述已自动生成**

**文本

描述已自动生成**

编写程序，向一个HTTP服务器提交一个表单：使用URLConnection类和第5章的QueryString类提交(post)表单数据。

具体要求：

1）向服务器资源发送名字”Elliotte Rusty Harold”和电子邮件地址elharo@biblio.org。

2）服务器资源位于http://www.cafeaulait.org/books/jnp4/postquery.html, 是一个表单测试器，可接受任何使用POST或GET方法的输入,并返回一个HTML页面,显示所提交的名和值。返回的数据是HTML

**文本

描述已自动生成**

在上面的代码中，我们创建了一个URL对象，指定要提交表单的HTTP服务器资源的URL。然后，我们使用openConnection方法打开与URL的连接，并将连接转换为HttpURLConnection对象。我们将请求方法设置为POST，并设置DoOutput为true，以允许发送表单数据。

我们构建了表单数据，并设置了请求头的Content-Type和Content-Length属性。然后，我们使用DataOutputStream将表单数据写入输出流，并发送到服务器。

通过运行这个程序，你将能够向HTTP服务器提交一个表单，并在控制台显示输出服务器返回的HTML页面。

**文本, 信件

描述已自动生成** **文本

描述已自动生成**

**图形用户界面, 文本, 电子邮件

描述已自动生成** **文本

描述已自动生成**

**图形用户界面, 文本, 应用程序

描述已自动生成** **图示, 文本

描述已自动生成**

* **断开连接可以关闭流**
* **关闭流不会断开连接**

**文本, 信件

描述已自动生成** **图示

描述已自动生成**

**图片包含 文本

描述已自动生成** **文本

描述已自动生成**

**图形用户界面, 文本, 应用程序

描述已自动生成** **文本

描述已自动生成**

文本

描述已自动生成 文本

描述已自动生成

**文本

描述已自动生成**

**文本

描述已自动生成**

在上面的代码中，你需要将senderEmail和senderPassword替换为你自己的邮箱帐号和密码。同时，将recipientEmail替换为接收邮件的邮箱地址。

你还需要将properties.put("mail.smtp.host", "smtp.example.com")中的smtp.example.com替换为你的SMTP服务器地址，将`properties.put("mail.smtp.port", "587")`中的`587`替换为你的SMTP服务器端口号。

通过运行这个程序，你将能够使用Java发送邮件到指定的邮箱地址。邮件的内容将包括学号、姓名和自定义的邮件发送程序信息。

请注意，为了成功发送邮件，你需要确保你的邮箱帐号开启了SMTP服务，并提供了正确的SMTP服务器地址和端口号。另外，你可能需要导入JavaMail API的相关库文件，以便程序能够正常编译和运行。

**密码**

图示

描述已自动生成 文本

中度可信度描述已自动生成

**文本

描述已自动生成**

**编写一个名为ClipherSuites的类，实现以下功能：生成安全socket (协议为https) ，端口为：443按行打印出当前socket所支持所有密码组并输出密码组的个数**

**文本

描述已自动生成**