```
/*
 * Created on Jul 7, 2004
 *
 */
package pony;

import java.security.*;
import java.util.Properties;
import javax.mail.*;

/**
 * @author rocha
 *
 */
public class MailReceiver
{

  static String URL = "imap://ca.rocha:password@imaptoo.media.mit.edu:993/INBOX/)";

  static String EMAIL = "goo-p@media.mit.edu";

  static int DELAY = 15; // seconds

  static boolean debug = true;

  static SerialControl serialControl;

  public static void main(String argv[]) throws Exception {
    System.out.println("Simplicity Iron Chef Contest");
    System.out.println("TEAM Lovely (Evil) Pony");
    System.out.println("GOO-PRINT version 1.0");
    System.out.println("-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-\n");

    // initiate comm
    System.out.println("Initiating COM...");
    serialControl = new SerialControl();

    // check mail
    while (true) {
      System.out.println("\nChecking E-MAIL");
      checkMail();
      Thread.sleep(DELAY * 1000);
    }
  }

  static void checkMail() throws Exception {
    Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
    // Security.setProperty( "ssl.SocketFactory.provider",
    // "DummySSLSocketFactory");

    final String SSL_FACTORY = "javax.net.ssl.SSLSocketFactory";

    // Get a Properties object
    Properties props = System.getProperties();
    props.setProperty("mail.imap.socketFactory.class", SSL_FACTORY);
    props.setProperty("mail.pop3.socketFactory.class", SSL_FACTORY);

    props.setProperty("mail.imap.socketFactory.fallback", "false");
    props.setProperty("mail.pop3.socketFactory.fallback", "false");

    Session session = Session.getInstance(props);
    //session.setDebug( debug);

    URLName urln = new URLName(URL);

    Store store = session.getStore(urln);
    store.connect();

    //Folder folder = store.getDefaultFolder();
    Folder folder = store.getFolder("INBOX");
    checkFolder(folder);
```

```
    //dumpRecursive(folder, true, "");
    store.close();
  }

  static void checkFolder(Folder folder) throws Exception {
    if ((folder.getType() & Folder.HOLDS_FOLDERS) != 0) {
      System.out.print("Directory: " + folder.getName());
    } else {
      System.out.print("Name: " + folder.getName());
    }
    System.out.print(" :: " + folder.getFullName());
    System.out.print(" :: " + folder.getURLName());
    System.out.println();

    if (!folder.isSubscribed()) {
      System.out.println("Not Subscribed");
    }

    if ((folder.getType() & Folder.HOLDS_MESSAGES) != 0) {
      System.out.print("Total Messages: " + folder.getMessageCount());
      System.out.print("  New Messages: " + folder.getNewMessageCount());
      System.out.println("  Unread Messages: "
          + folder.getUnreadMessageCount());

      if (folder.hasNewMessages()) {
        checkMessagges(folder);
      }

    }
    System.out.println();
  }

  private static void checkMessagges(Folder folder) throws Exception {
    folder.open(Folder.READ_WRITE);

    Message[] messages = folder.getMessages();
    int count = messages.length;

    if (count == 0) {
      System.out.println("no messages");
      return;
    }

    for (int i = 0; i < count; i++) {
      Message msg = messages[i];
      if (checkAddress(msg)) {
        // mark it as read, so it doesnt show up next time
        msg.setFlag(Flags.Flag.SEEN, true);

        // that means i got to print this baby out
        System.out.println(".........................");
        System.out.print("FROM: ");
        System.out.println(msg.getFrom()[0]);
        System.out.print("SUBJECT: ");
        System.out.println(msg.getSubject());
        System.out.println();
        System.out.println("");
        System.out.println(msg.getContent());
        System.out.println(".........................");

        // send string to printer
        System.out.println("Printing Message...");
        serialControl.writeString((String) msg.getContent());
        System.out.println("Done");
        break;
      }
    }

    folder.close(false);
  }
```

```
  private static boolean checkAddress(Message message)
      throws MessagingException {
    Address[] recipients = message.getRecipients(Message.RecipientType.TO);
    int count = recipients.length;
    for (int i = 0; i < count; i++) {

      // check if it is new
      Flags flags = message.getFlags();
      if (flags.contains(Flags.Flag.SEEN)) {
        // ignore if it is not new
        continue;
      }

      // check it is the proper recipient
      String recipient = recipients[i].toString();
      if (recipient.equalsIgnoreCase(EMAIL)) {
        return true;
      }
    }
    return false;
  }
}
```