

CSCI 415: Computer and Network Security

Lab 4: Introduction to some basic network and security concepts in Java (50 points)

Due: 10/9/09 by 11:59 PM

Goals

- to introduce students to some basic network and security concepts in Java (i.e. who needs “whois.com”, “ipconfig”, or “Shieldsup” When you can create your own.)

Introduction

- Some relevant and exciting core packages of Java are java.net and java.io which contain a number of useful classes that allow network programmers to do cool stuff
- Using many of these classes will allow us to communicate with any server on the Internet or implement our own Internet server
- Over the semester we will use sockets to program TCP and UDP as well as Mail clients, and we will also look at network programming with GUIs – as we cover the many protocols
- The [java.io package](#) contains classes that support input and output
- The [java.net package](#) provides classes for implementing networking applications
-

4a) Who Needs WHOIS.com?

- The InetAddress class and the GetByName Method
 - The [InetAddress class](#) represents an Internet Protocol (IP) address
 - This class handles Internet addresses both as host names and as IP addresses
 - The [static method getByName](#) of this class uses DNS (Domain Name System) to return the Internet address of a specified host name as an InetAddress object
 - getByName method:
 - **Parameters:**
 - host or null
 - **Returns:**
 - an IP address for the given host name.
 - **Throws:**
 - [UnknownHostException](#) - if no IP address for the host could be found

```
// CSCI 415
```

```
_____;  
import java.util.Scanner;  
import java.io.*;  
  
public class IPFinder {  
    public static void main(String[] args) throws IOException {  
        String _____;  
        Scanner input = new Scanner(System.in);  
  
        System.out.print("Enter host name: ");  
        host = input.nextLine();  
  
        try {  
            InetAddress address = InetAddress._____(_____);  
            System.out.println("IP address: " + address.toString());  
        }  
        _____ (UnknownHostException e) {  
            System.out.println("Could not find " + host);  
        }  
    }  
}
```

```
}  
}
```

Fill in the blanks

4b) Who Needs Ipconfig?

- The InetAddress class and the getLocalHost method
 - The [static method getLocalHost](#) of this class returns local host
 - getLocalHost method:
 - **Returns:**
 - the IP address of the local host.
 - **Throws:**
 - [UnknownHostException](#) - if no IP address for the host could be found.

```
○  
// CSCI 415  
import java.net.*;  
  
public class MyLocalIPAddress {  
    public static void main(String[] args) {  
        try {  
            _____ address = InetAddress._____( );  
            _____ (address);  
        }  
        catch ( _____ ) {  
            System.out.println(" _____ ");  
        }  
    }  
}
```

Fill in the blanks

4c) Who needs ShieldsUp?

- This program checks on the range of ports on a specified host and reports back on the ports that are offering service
 - The program tries to create a socket on each port number in turn
 - If a socket is created successfully then it means that there is an open socket – otherwise an IOException is thrown

```
/*CSCI 415  
 * this program looks up a range of ports that are providing service on a specified host  
 * Student Name _____  
 * Date _____  
 */  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
_____  
_____;
```

```
public class PortScanner extends JFrame implements ActionListener {  
  
    private JLabel prompt;  
    private JTextField hostInput;  
    private JTextArea report;  
    private JButton seekButton, exitButton;  
    private JPanel hostPanel, buttonPanel;  
    private static Socket socket = null;  
  
    public static void main(String[] args) throws IOException {
```

```

PortScanner scanner = _____ PortScanner();
scanner.setSize(400, 300);
scanner.setVisible(_____); // Boolean value

scanner.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {

        // check to see if socket is open

        if (socket != _____)
            try {
                socket.close();
            } catch (IOException ioEx) {
                System.out.println("\n ~ sorry, cannot close link ~");
            }

        System.exit(0);
    }
});
}

public _____() {

    Container pane = getContentPane();
    hostPanel = new JPanel();
    prompt = new JLabel("Host name: ");
    hostInput = new JTextField("enter host name here", 25); // test with cs.millersville.edu and

```

cs.mit.edu

```

hostPanel.add(prompt);
hostPanel.add(hostInput);
pane.add(hostPanel, BorderLayout.NORTH);
report = new JTextArea(10, 25);
pane.add(report, BorderLayout.CENTER);
buttonPanel = new JPanel();
seekButton = new JButton("lookup ports providing service");
seekButton.addActionListener(this);
buttonPanel.add(seekButton);
exitButton = new JButton("exit");
exitButton.addActionListener(this);
buttonPanel.add(exitButton);
pane.add(buttonPanel, BorderLayout.SOUTH);
}

public void actionPerformed(ActionEvent event) {

    if (event.getSource() == exitButton)
        System.exit(0);
    report.setText("");

    // retrieve this URL
    String host = hostInput.getText();

    try {
        // convert URL string into an InetAddress

        InetAddress theAddress = InetAddress._____ (_____);
        report.append("IP address: " + theAddress + "\n");
        for (_____ {
            try {

```

