Network Programming

Network Interface in java

What is a Network Interface?

- A network interface can be thought of as a point at which your computer connects to the network.
- It is not necessarily a piece of hardware but can also be implemented in software.

What is a Network Interface?

- In everyday language, we refer to them by the term Network Interface Cards (NICs) – but they don't all have to be of hardware form.
- For example, the popular localhost IP 127.0.0.1
  - which we use a lot in testing web and network applications is the loopback interface – which is not a direct hardware interface.

What is a Network Interface?

- Of course, systems often have multiple active network connections, such as wired Ethernet, WIFI, Bluetooth, etc.

Java.net.NetworkInterface class in Java

- In Java, the main API we can use to interact directly with Network Interface is the *java.net.NetworkInterface* class.

Java.net.NetworkInterface class in Java

- This class represents network interface, both software as well as hardware, its name, list of IP addresses assigned to it, and all related information.

Java.net.NetworkInterface class in Java

- This class can be used in cases when we want to specifically use a particular interface for transmitting our packet on a system with multiple NICs.
  - For example, a loopback interface which is used for testing purposes.

Java.net.NetworkInterface class in Java

- Since NetworkInterface objects represent physical hardware and virtual addresses, they cannot be constructed arbitrarily.
- As with the InetAddress class, there are static factory methods that return the NetworkInterface object associated with a particular network interface.

Java.net.NetworkInterface class in Java

- You can ask for a NetworkInterface by IP address, by name, or by enumeration.

Java.net.NetworkInterface class (Methods)

1.getName() : Returns the name of this network interface.

```
Syntax : public String getName()
```

Java.net.NetworkInterface class (Methods)

2.getInetAddresses() : Returns an enumeration of all Inetaddresses bound to this network interface, if security manager allows it.

```
Syntax :public Enumeration getInetAddresses()
```

Java.net.NetworkInterface class (Methods)

3.getInterfaceAddresses() : Returns a list of all interface addresses on this interface.

```
Syntax :public List getInterfaceAddresses()
```

Java.net.NetworkInterface class (Methods)

4.getSubInterfaces() : Returns an enumeration of all the sub or virtual interfaces of this network interface. For example, eth0:2 is a sub interface of eth0.

```
Syntax :public Enumeration getSubInterfaces()
```

Java.net.NetworkInterface class (Methods)

5.getParent() : In case of a sub interface, this method returns the parent interface. If this is not a subinterface, this method will return null.

```
Syntax :public NetworkInterface getParent()
```

Java.net.NetworkInterface class (Methods)

6.getIndex() : Returns the index assigned to this network interface by the system. Indexes can be used in place of long names to refer to any interface on the device.

```
Syntax :public int getIndex()
```

Java.net.NetworkInterface class (Methods)

7.getDisplayName() : This method returns the name of network interface in a readable string format.

```
Syntax :public String getDisplayName()
```

Java.net.NetworkInterface class (Methods)

8.getByName() : Finds and returns the network interface with the specified name, or null if none exists.

```
Syntax :public static NetworkInterface getByName(String name)
                                        throws SocketException

Parameters :

name : name of network interface to search for.

Throws :

SocketException : if I/O error occurs.
```

Java.net.NetworkInterface class (Methods)

9.getByIndex() : Performs similar function as the previous function with index used as search parameter instead of name.

```
Syntax :public static NetworkInterface getByIndex(int index)
                                        throws SocketException
Parameters :
index : index of network interface to search for.
Throws :
SocketException : if I/O error occurs.
```

Java.net.NetworkInterface class (Methods)

10.getByInetAddress() : This method is widely used as it returns the network interface the specified inetaddress is bound to. If an InetAddress is bound to multiple interfaces, any one of the interfaces may be returned.

Syntax : public static NetworkInterface getByInetAddress(InetAddress addr)

throws SocketException

Parameters :

addr : address to search for

Throws :

SocketException : If IO error occurs

Java.net.NetworkInterface class (Methods)

11.getNetworkInterfaces() : Returns all the network interfaces on the system.

```
Syntax :public static Enumeration getNetworkInterfaces()
                                                        throws
SocketException
Throws :
SocketException : If IO error occurs
```

Java.net.NetworkInterface class (Methods)

12.isUp() : Returns a boolean value indicating if this network interface is up and running.

```
Syntax : public boolean isUp()
```

Java.net.NetworkInterface class (Methods)

13.isLoopback() : Returns a boolean value indicating if this interface is a loopback interface or not.

```
Syntax : public boolean isLoopback()
```

Java.net.NetworkInterface class (Methods)

14.isPointToPoint() : Returns a boolean value indicating if this interface is a point to point interface or not.

```
Syntax : public boolean isPointToPoint()
```

Java.net.NetworkInterface class (Methods)

15.supportsMulticast() : Returns a boolean value indicating if this interface supports multicasting or not.

```
Syntax : public boolean supportsMulticast()
```

Java.net.NetworkInterface class (Methods)

16.getHardwareAddress() : Returns a byte array containing the hardware address(MAC) address of this interface. The caller must have appropriate permissions before calling this method.

```
public byte[] getHardwareAddress()
```

Java.net.NetworkInterface class (Methods)

- **17.getMTU()** : Returns the maximum transmission unit of this interface. An MTU is the largest size of the packet or frame that can be sent in packet based network.

```
Syntax :public int getMTU()
```

Java.net.NetworkInterface class (Methods)

- **18.isVirtual()** : Returns a boolean value indicating whether this interface is a virtual interface or not. Virtual interfaces are used in conjunction physical interfaces to provide additional values such as addresses and MTU.

```
Syntax : public boolean isVirtual()
```

Java.net.NetworkInterface class (Methods)

- **19.equals()** : This method is used to compare two network interfaces for equality. Two network interfaces are equal if they have same name and addresses bound to them.

```
Syntax :public boolean equals(Object obj)

Parameters :

obj : Object to compare this network interface for equality
```

Java.net.NetworkInterface class (Methods)

- **20.hashCode()** : Returns the hashcode value for this object.

```
Syntax :public int hashCode()
```

Java.net.NetworkInterface class (Methods)

- **21.toString()** : Returns a textual description of this object.

```
Syntax :public String toString()
```

Java.net.NetworkInterface class (Java Implementation)

```
//Java program to illustrate various
//networkInterface class methods.
import java.net.InetAddress;
import java.net.InterfaceAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Enumeration;

public class NetworkInterfaceEx
{
    public static void main(String[] args) throws SocketException,
                                                   UnknownHostException
    {
```

Java.net.NetworkInterface class (Java Implementation)

```
class NetworkInterfaceEx

lic static void main(String[] args) throws SocketException,
                                           UnknownHostException


 // getNetworkInterfaces() returns a list of all interfaces
 // present in the system.
 ArrayList<NetworkInterface> interfaces = Collections.list(
                                      NetworkInterface.getNetworkInterfaces());

 System.out.println("Information about present Network Interfaces...\n");
 for (NetworkInterface iface : interfaces)
 {
     // isUp() method used for checking whether the interface in process
     // is up and running or not.
     if (iface.isUp())
     {

         // getName() method
         System.out.println("Interface Name: " + iface.getName());

         // getDisplayName() method
         System.out.println("Interface display name: " + iface.getDisplayName());
```

Java.net.NetworkInterface class (Java Implementation)

```
// gethardwareaddress() method
System.out.println("Hardware Address: " +
                   Arrays.toString(iface.getHardwareAddress()));

// getParent() method
System.out.println("Parent: " + iface.getParent());

// getIndex() method
System.out.println("Index: " + iface.getIndex());
// Interface addresses of the network interface
System.out.println("\tInterface addresses: ");

// getInterfaceAddresses() method
for (InterfaceAddress addr : iface.getInterfaceAddresses())
{
    System.out.println("\t\t" + addr.getAddress().toString());
}
// Interface addresses of the network interface
System.out.println("\tInetAddresses associated with this interface: ");
```

Java.net.NetworkInterface class (Java Implementation)

```
// getInetAddresses() method returns list of all
// addresses currently bound to this interface
Enumeration<InetAddress> en = iface.getInetAddresses();
while (en.hasMoreElements())
{
    System.out.println("\t\t" + en.nextElement().toString());
}

// getMTU() method
System.out.println("\tMTU: " + iface.getMTU());

// getSubInterfaces() method
System.out.println("\tSubinterfaces: " +
                   Collections.list(iface.getSubInterfaces()));

// isLoopback() method
System.out.println("\this loopback: " + iface.isLoopback());

// isVirtual() method
System.out.println("\this virtual: " + iface.isVirtual());

// isPointToPoint() method
System.out.println("\this point to point: " + iface.isPointToPoint());
```

Java.net.NetworkInterface class (Java Implementation)

```
        // supportsMulticast() method
        System.out.println("Supports Multicast: " + iface.supportsMulticast());

    }
}

// getByIndex() method returns network interface
// with the specified index
NetworkInterface nif = NetworkInterface.getByIndex(1);

// toString() method is used to display textual
// information about this network interface
System.out.println("Network interface 1: " + nif.toString());
```

Java.net.NetworkInterface class (Java Implementation)

```
        // getByName() method returns network interface
        // with the specified name
        NetworkInterface nif2 = NetworkInterface.getByName("eth0");
        InetAddress ip = InetAddress.getByName("localhost");

        // getbyInetAddress() method
        NetworkInterface nif3 = NetworkInterface.getByInetAddress(ip);
        System.out.println("\nlocalhost associated with: " + nif3);

        // equals() method
        boolean eq = nif.equals(nif2);
        System.out.println("nif==nif2: " + eq);

        // hashCode() method
        System.out.println("Hashcode : " + nif.hashCode());
    }
}
```

Java.net.NetworkInterface class (Java Implementation)

- Output :

```
Information about present Network Interfaces...

Interface Name: lo
Interface display name: Software Loopback Interface 1
Hardware Address: null
Parent: null
Index: 1
    Interface addresses:
        /127.0.0.1
        /0:0:0:0:0:0:0:1
    InetAddresses associated with this interface:
        /127.0.0.1
        /0:0:0:0:0:0:0:1
    MTU: -1
    Subinterfaces: []
    is loopback: true
    is virtual: false
    is point to point: false
Supports Multicast: true
Interface Name: wlan5
Interface display name: Dell Wireless 1705 802.11b|g|n (2.4GHZ)
Hardware Address: [100, 90, 4, -90, 2, 15]
Parent: null
```

Java.net.NetworkInterface class (Java Implementation)

- Output :

```
Index: 16
    Interface addresses:
        /192.168.43.96
        /2405:205:1486:9a1b:e567:b46f:198a:fe0c
        /2405:205:1486:9a1b:8c93:9f82:6dd2:350c
        /fe80:0:0:0:e567:b46f:198a:fe0c%wlan5
    InetAddresses associated with this interface:
        /192.168.43.96
        /2405:205:1486:9a1b:e567:b46f:198a:fe0c
        /2405:205:1486:9a1b:8c93:9f82:6dd2:350c
        /fe80:0:0:0:e567:b46f:198a:fe0c%wlan5
    MTU: 1500
    Subinterfaces: []
    is loopback: false
    is virtual: false
    is point to point: false
Supports Multicast: true
Network interface 1: name:lo (Software Loopback Interface 1)

localhost associated with: name:lo (Software Loopback Interface 1)
nif==nif2: false
HashCode : 2544
```