Process and Service Programming

4.2 Auxiliary classes for networking

I.E.S. Doctor Balmis

4.2 Auxiliary classes for networking

4.2.1. java.net.NetworkInterface

This class represents network interface, both software as well as hardware, its name, list of IP addresses assigned to it and all related information. It can be used in cases when we want to specifically use a particular interface for transmitting our packet on a system with multiple NICs.

> java.net.InetAddress specification
> (https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/NetworkInterface.html)

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 a software. For example a loopback interface which is used for testing purposes.

Method	Description
public static Enumeration getNetworkInterfaces()	Returns all the network interfaces on the system.
public List getInterfaceAddresses()	Returns a list of all interface addresses on this interface.
public Enumeration getInetAddresses()	Returns an enumeration of all InetAddresses bound to this network interface, if security manager allows it.
public String getName()	Returns the name of this network interface.
public int 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.
public String getDisplayName()	This method returns the name of network interface in a readable string format.
public static NetworkInterface getByName(String name)	Finds and returns the network interface with the specified name, or null if none exists.
public static NetworkInterface getByIndex(int index)	Performs similar function as the previous function with index used as search parameter instead of name.
public static NetworkInterface getByInetAddress(InetAddress addr)	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.
public boolean isUp()	Returns a boolean value indicating if this network interface is up and running.

java
```java
// Java program to illustrate various java.net.NetworkInterface class methods.

public class NetworkInterfaceExample
{
    public 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());

                // 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: ");

                // 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("\tis loopback: " + iface.isLoopback());

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

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

                // 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());

        // 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);
    }
}

```

4.2.2 java.net.InterfaceAddress

This class represents a network interface address. Every device that has an IP address has an IP address on the network interface.

In short it's an IP address, a subnet mask and a broadcast address when the address is an IPv4 one. An IP address and a network prefix length in the case of IPv6 address.

> java.net.InterfaceAddress specification (https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/InterfaceAddress.html)

Method	Description
public InetAddress getAddress()	Returns an InetAddress for this address.
public InetAddress getBroadcast()	Returns the InetAddress for the broadcast address for this interface address. As only IPv4 addresses have broadcast addresses, null would be returned on using an IPv6 address.
public short getNetworkPrefixLength()	Returns the prefix length for this interface address, i.e. subnet mask for this address.

```java
// Java program to illustrate methods of java.net.InterfaceAddress class

public class InterfaceaddressExample
{
    public static void main(String[] args) throws SocketException
    {
        // Modify according to your system
        NetworkInterface nif = NetworkInterface.getByIndex(1);
        List<InterfaceAddress> list = nif.getInterfaceAddresses();

        for (InterfaceAddress iaddr : list)
        {
            // getAddress() method
            System.out.println("getAddress() : " + iaddr.getAddress());

            // getBroadcast() method
            System.out.println("getBroadcast() : " + iaddr.getBroadcast());

            // getNetworkPrefixLength() method
            System.out.println("PrefixLength : " + iaddr.getNetworkPrefixLength());

            // hashCode() method
            System.out.println("Hashcode : " + iaddr.hashCode());

            // toString() method
            System.out.println("toString() : " + iaddr.toString());
        }
    }
}

```

4.2.3. java.net.InetAddress

Java InetAddress class represents an IP address. The java.net.InetAddress class provides methods to get the IP of any host name for example www.google.com, www.facebook.com, etc.

> java.net.InetAddress specification (https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/InetAddress.html)

An instance of InetAddress represents the IP address with its corresponding host name. There are two types of addresses: Unicast and Multicast. The Unicast is an identifier for a single interface whereas Multicast is an identifier for a set of interfaces.

Local Name Resolver (hosts file)

You should know that DNS translates domain names like into IP addresses. But did you know that there's a file on your system that can override that? It's called your hosts file and lets you map specific domain names to an IP address of your choosing. Your HOSTS file only affects your computer, so you can use it to create custom URLs for IP addresses on your network, or you can use it to redirect certain websites.

As you can imagine, editing the HOSTS file can easily break your internet if it's modified incorrectly or maliciously. So, it's not particularly easy for a normal user to edit. This is a good thing.

- Windows

The HOSTS file is normally stored in a plain text file in the Windows System folder.

Hit the start menu or press the Windows key and start typing Notepad.

Right-click Notepad and choose Run as administrator.

In Notepad, click File then Open... In the File name field, paste the following path in:

c:\Windows\System32\Drivers\etc\hosts

Now you'll be able to edit and save changes to your HOSTS file.

To map a domain, add a line based on the examples in the HOSTS file.

- OS X & GNU/Linux

The file is in /etc/hosts and you should edit it with administrator privileges.

```sh
# Add the following lines to the hosts file
## At school
##  - use your computer IP for the 'cliente' and 'servidor' entries.
##  - use the teacher's computer IP as the 'profesor' entry
## At home
##  - use your computer IP for the 'cliente', 'servidor' and 'profesor' entries.

# In all the activities, we will always use these domain names
# making our apps work at home and at school without having to change any IP address.
10.100.XX.1 cliente.psp
10.100.XX.1 servidor.psp
10.100.0.1 profesor.psp
```

Method	Description
public static InetAddress getByName(String host) throws UnknownHostException	It returns the instance of InetAddress containing LocalHost IP and name.
public static InetAddress getLocalHost() throws UnknownHostException	It returns the instance of InetAdddress containing local host name and address.
public String getHostName()	It returns the host name of the IP address.
public String getHostAddress()	It returns the IP address in string format.
public boolean isReachable(int timeout)	This method tests whether that address is reachable.

```java

class InetAddressExample {
    public static void main(String[] args)
        throws UnknownHostException
    {
        // To get and print InetAddress of Local Host
        InetAddress address1 = InetAddress.getLocalHost();
        System.out.println("InetAddress of Local Host : "
                        + address1);

        // To get and print InetAddress of Named Host
        InetAddress address2
            = InetAddress.getByName("45.22.30.39");
        System.out.println("InetAddress of Named Host : "
            + address2);

    // To get and print ALL InetAddresses of Named Host
    InetAddress address3[]
      = InetAddress.getAllByName("172.19.25.29");
    for (int i = 0; i < address3.length; i++) {
      System.out.println(
        "ALL InetAddresses of Named Host : "
        + address3[i]);
    }

    // To get and print InetAddresses of
    // Host with specified IP Address
    byte IPAddress[] = { 125, 0, 0, 1 };
    InetAddress address4
      = InetAddress.getByAddress(IPAddress);
    System.out.println(
      "InetAddresses of Host with specified IP Address : "
      + address4);

    // To get and print InetAddresses of Host
    // with specified IP Address and hostname
    byte[] IPAddress2
      = { 105, 22, (byte)223, (byte)186 };
    InetAddress address5 = InetAddress.getByAddress(
      "gfg.com", IPAddress2);
    System.out.println(
      "InetAddresses of Host with specified IP Address and hostname : "
      + address5);
  }
}
```

Host seeker (U4S4_HostSeeker)

Your computer is connected to a LAN (Local Area Network) and probably it is using private IP addresses.

Addresses can be one of class C (192.168.X.Y), class B (172.17.X.Y) or class A (10.X.Y.Z). That depends on the network mask or network prefix used for the network interface configuration.

You can also check it using Linux **ifconfig** command or Windows **ipconfig** command.

Write a program to know which hosts are up and running in your network, that is, which hosts are reachable from you computer by using one of the interfaces.

First, you can write specific code to test your network. Once your app is working, try to make it generic and reusable by making it work in any network, detecting the network prefix and checking all the possible IPs in a network.

The app will get a Network interface card name as argument and will check only the IPv4 addresses attached to that interface. We can know if an IP is IPv4 or IPv6 using the operator `instanceof` with Inet4Address and Inet6Address subclasses of InetAddress

4.2.4 java.net.URL

The Java URL class represents an URL. URL is an acronym for Uniform Resource Locator.

> java.net.URL specification (https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/URL.html)

It points to a resource on the World Wide Web. For example:

> http://psp2dam.github.io/psp_pages/

A URL contains many information:

- **Protocol**: In this case, `http` is the protocol.
- **Server name or IP Address**: In this case, `psp2dam.github.io` is the server name.
- **Port Number**: It is an optional attribute. Many times it is derived from the protocol, by chosing its standard default port. In the example the port is missing but it is set to `80`.
- **File Name or directory name**: In this case, only the path (directory) is specified in the URL. Depending on the server configuration the file name can take a default value. In the example `index.html` is the file name.

Constructor	Description
URL(String spec)	Creates an instance of a URL from the String representation.
URL(String protocol, String host, int port, String file)	Creates an instance of a URL from the given protocol, host, port number, and file.
URL(String protocol, String host, int port, String file, URLStreamHandler handler)	Creates an instance of a URL from the given protocol, host, port number, file, and handler.
URL(String protocol, String host, String file)	Creates an instance of a URL from the given protocol name, host name, and file name.
URL(URL context, String spec)	Creates an instance of a URL by parsing the given spec within a specified context.
URL(URL context, String spec, URLStreamHandler handler)	Creates an instance of a URL by parsing the given spec with the specified handler within a given context.

The java.net.URL class provides many methods. The important methods of URL class are given below.

Method	Description
public String getProtocol()	it returns the protocol of the URL.
public String getHost()	it returns the host name of the URL.
public String getPort()	it returns the Port Number of the URL.
public String getFile()	it returns the file name of the URL.
public String getAuthority()	it returns the authority of the URL.
public String toString()	it returns the string representation of the URL.
public String getQuery()	it returns the query string of the URL.
public String getDefaultPort()	it returns the default port of the URL.
public URLConnection openConnection()	it returns the instance of URLConnection i.e. associated with this URL.
public InputStream openStream()	it opens a connection to this URL and returns an InputStream for reading from that connection.

Method	Description
public boolean equals(Object obj)	it compares the URL with the given object.
public Object getContent()	it returns the content of the URL.
public String getRef()	it returns the anchor or reference of the URL.
public URI toURI()	it returns a URI of the URL.

```java
//URLDemo.java
public static void main(String[] args) throws MalformedURLException{

    URL url=new URL("http://psp2dam.github.io/psp_pages");

    System.out.println("Protocol: "+url.getProtocol());
    System.out.println("Host Name: "+url.getHost());
    System.out.println("Port Number: "+url.getPort());
    System.out.println("File Name: "+url.getFile());
}
```

Let us see another example URL class in Java.

```java
//URLDemo.java
public static void main(String[] args){
    URL url=new URL("https://www.google.com/search?q=javatpoint&oq=javatpoint&sourceid=chrome&ie=UTF-8");

    System.out.println("Protocol: "+url.getProtocol());
    System.out.println("Host Name: "+url.getHost());
    System.out.println("Port Number: "+url.getPort());
    System.out.println("Default Port Number: "+url.getDefaultPort());
    System.out.println("Query String: "+url.getQuery());
    System.out.println("Path: "+url.getPath());
    System.out.println("File: "+url.getFile());
}
```

```sh
Protocol: https
Host Name: www.google.com
Port Number: -1
Default Port Number: 443
Query String: q=javatpoint&oq=javatpoint&sourceid=chrome&ie=UTF-8
Path: /search
File: /search?q=javatpoint&oq=javatpoint&sourceid=chrome&ie=UTF-8
```

4.2.5 java.net.URLConnection

The Java URLConnection class represents a communication link between the URL and the application. It can be used to read and write data to the specified resource referred by the URL.

> java.net.URLConnection specification (https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/URLConnection.html)

URLConnection is an abstract class. The two subclasses `HttpURLConnection` and `JarURLConnection` makes the connection between the client Java program and URL resource on the internet.

With the help of URLConnection class, a user can read and write to and from any resource referenced by an URL object. Once a connection is established and the Java program has an URLConnection object, we can use it to read or write or get further information like content length, etc.

The URLConnection class provides many methods. We can display all the data of a webpage by using the getInputStream() method. It returns all the data of the specified URL in the stream that can be read and displayed.

Method	Description
void connect()	It opens a communications link to the resource referenced by this URL, if such a connection has not already been established.
Object getContent()	It retrieves the contents of the URL connection.
String getContentEncoding()	It returns the value of the content-encoding header field.
int getContentLength()	It returns the value of the content-length header field.
long getContentLengthLong()	It returns the value of the content-length header field as long.
String getContentType()	It returns the value of the date header field.
long getDate()	It returns the value of the date header field.
boolean getDoInput()	It returns the value of the URLConnection's doInput flag.
boolean getDoInput()	It returns the value of the URLConnection's doOutput flag.
String getHeaderField(int n)	It returns the value of nth header field
String getHeaderField(String name)	It returns the value of the named header field.
String getHeaderFieldKey(int n)	It returns the key for the nth header field.
Map<String, List<String>> getHeaderFields()	It returns the unmodifiable Map of the header field.
long getIfModifiedSince()	It returns the value of the object's ifModifiedSince field.
InputStream getInputStream()	It returns an input stream that reads from the open condition.
long getLastModified()	It returns the value of the last-modified header field.
OutputStream getOutputStream()	It returns an output stream that writes to the connection.
URL getURL()	It returns the value of the URLConnection's URL field.
void setDoInput(boolean doinput)	It sets the value of the doInput field for this URLConnection to the specified value.
void setDoOutput(boolean dooutput)	It sets the value of the doOutput field for the URLConnection to the specified value.

How to get the object of URLConnection Class

The openConnection() method of the URL class returns the object of URLConnection class.

```java
// URLConnectionExample
public static void main(String[] args) throws MalformedURLException, IOException{

    // Creating an object of URL class

    // Custom input URL is passed as an argument
    URL u = new URL("www.google.com");

    // Creating an object of URLConnection class to
    // communicate between application and URL
    URLConnection urlconnect = u.openConnection();

    // Creating an object of InputStream class
    // for our application streams to be read
    InputStream stream
        = urlconnect.getInputStream();

    BufferedReader in =
        new BufferedReader(
            new InputStreamReader(stream));
    // Till the time URL is being read
    String line;
    while ((line = in.readLine()) != null) {

        // Continue printing the stream
        System.out.println(line);
    }
}
```

MalformedURLException

If you test the previous code, you will get a `MalformedURLException` exception. What should you change to make it work?

Images downloader (U4S7_ImagesDownloader)

Create an application to download images from a URL. The image URL must be given as an application argument and the image has to be saved in a images folder on the root of your project.