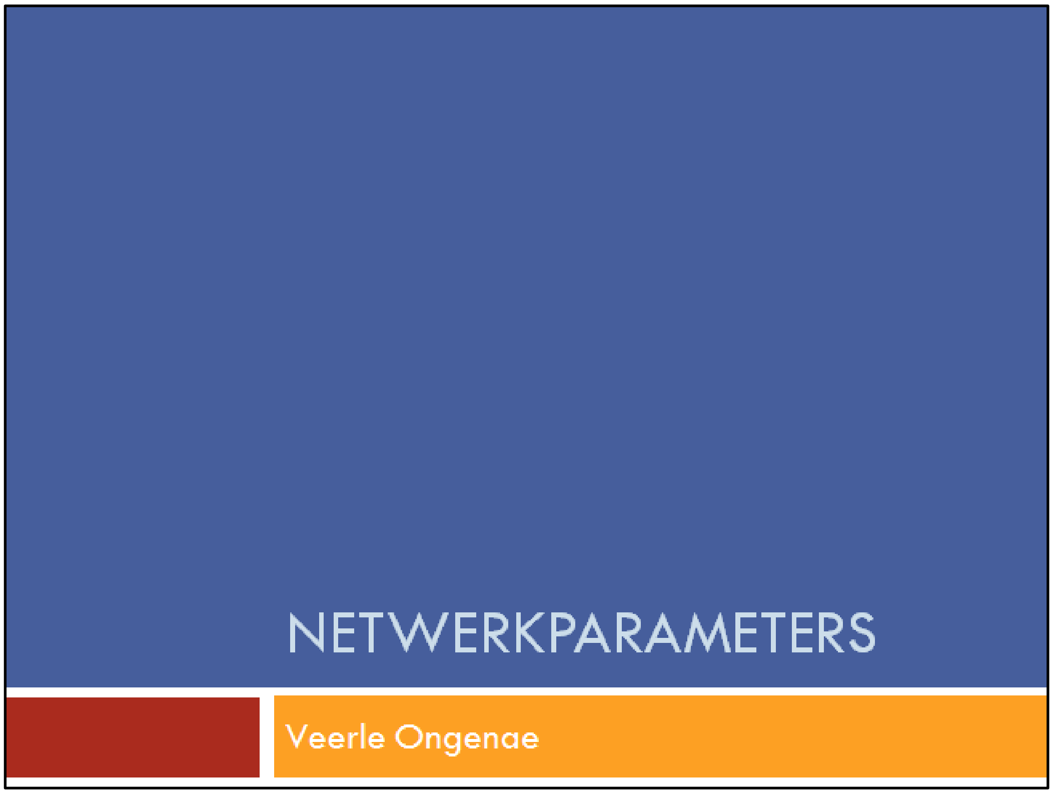
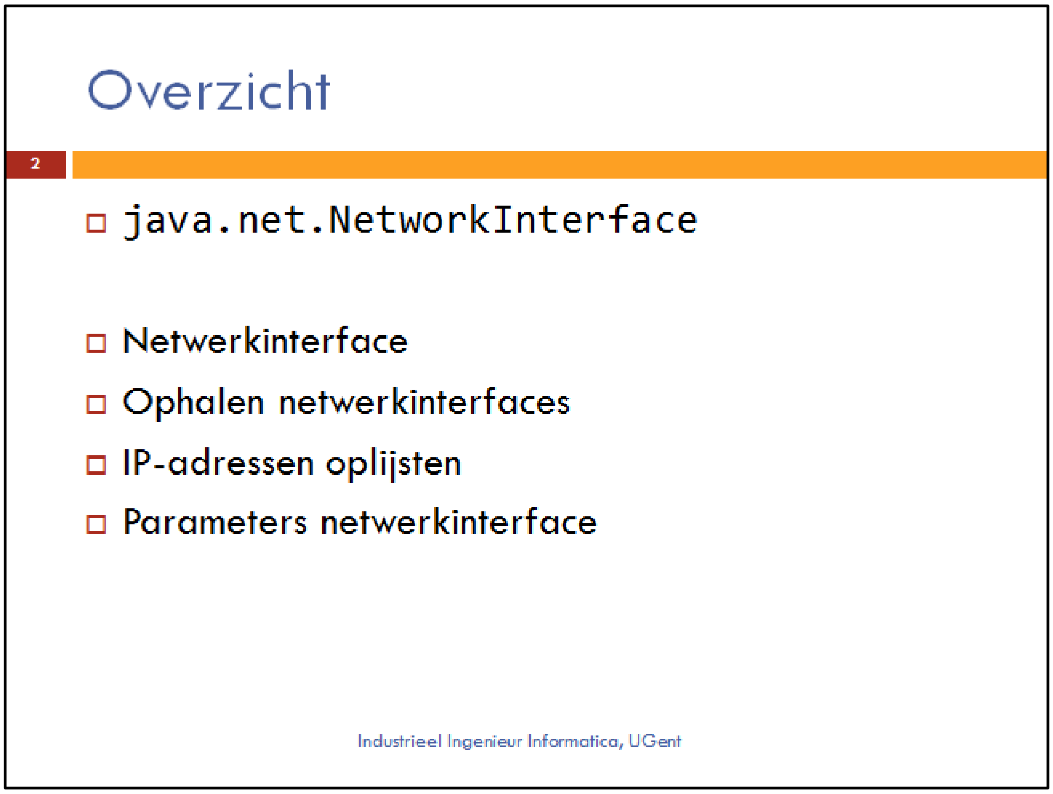


NETWERKPARAMETERS

Veerle Ongenae



Overzicht

2

- `java.net.NetworkInterface`
- Netwerkinterface
- Ophalen netwerkinterfaces
- IP-adressen oplijsten
- Parameters netwerkinterface

Industrieel Ingenieur Informatica, UGent

Systems often run with multiple active network connections, such as wired Ethernet, 802.11 b/g (wireless), and bluetooth. Some applications might need to access this information to perform the particular network activity on a specific connection.

The java.net.NetworkInterface class provides access to this information.

This lesson guides you through some of the more common uses of this class and provides examples that list all the network interfaces on a machine as well as their IP addresses and status.

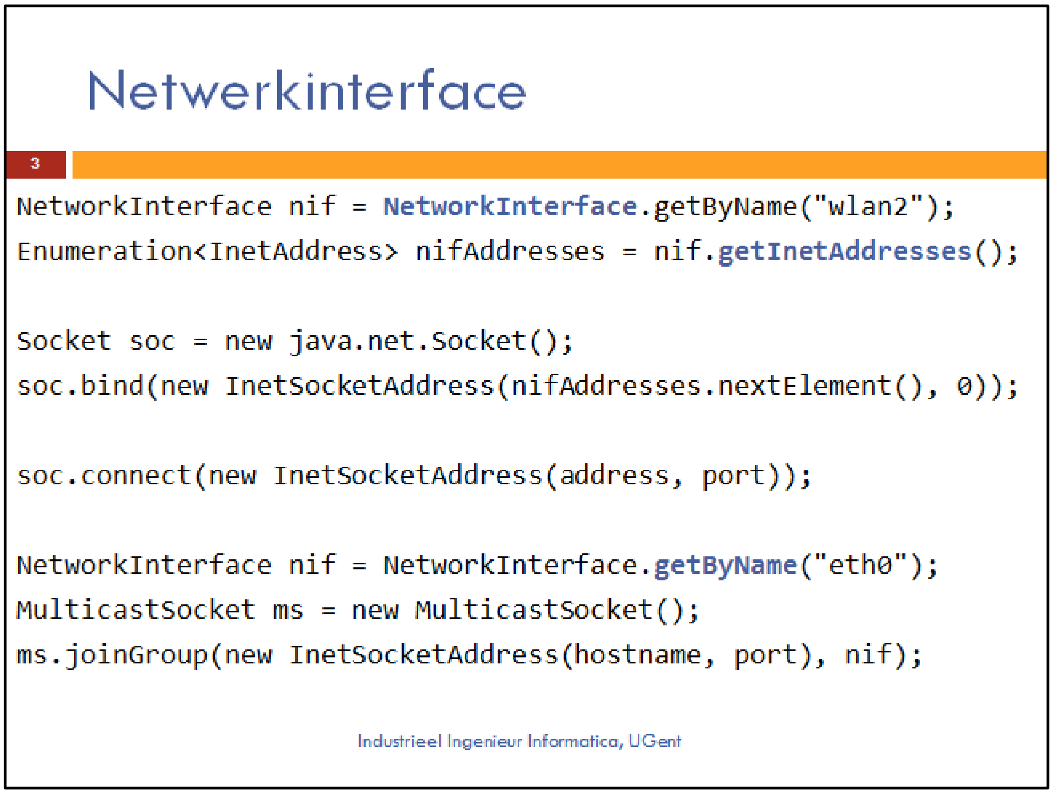
What Is a Network Interface?
This page describes a network interface and explains why you might want to use it.

Retrieving Network Interfaces
This page contains an example that illustrates how a client program can retrieve all the network interfaces on a machine.

Listing Network Interface Addresses
This page shows you how to list the IP addresses assigned to all the network interfaces on a machine.

Network Interface Parameters
This page shows you how to determine whether a network interface is running or if

the network interface is a loopback interface, a point-to-point interface, or a virtual interface. You can also learn how to determine if the interface supports multicasting.



Netwerkinterface

3

```
NetworkInterface nif = NetworkInterface.getByName("wlan2");
Enumeration<InetAddress> nifAddresses = nif.getInetAddresses();

Socket soc = new java.net.Socket();
soc.bind(new InetSocketAddress(nifAddresses.nextElement(), 0));

soc.connect(new InetSocketAddress(address, port));

NetworkInterface nif = NetworkInterface.getByName("eth0");
MulticastSocket ms = new MulticastSocket();
ms.joinGroup(new InetSocketAddress(hostname, port), nif);
```

Industrieel Ingenieur Informatica, UGent

What Is a Network Interface?
A *network interface* is the point of interconnection between a computer and a private or public network. A network interface is generally a **network interface card** (NIC), but does not have to have a **physical** form. Instead, the network interface can be implemented in software. For example, the loopback interface (127.0.0.1 for IPv4 and ::1 for IPv6) is not a physical device but a piece of software **simulating** a network interface. The loopback interface is commonly used in test environments.

The java.net.NetworkInterface class represents both types of interfaces.

NetworkInterface is useful for a multi-homed system, which is a system with multiple NICs. Using NetworkInterface, you can specify which NIC to use for a particular network activity.

For example, assume you have a machine with two configured NICs, and you want to send data to a server. You create a socket like this:
Socket soc = new java.net.Socket();
soc.connect(new InetSocketAddress(address, port));

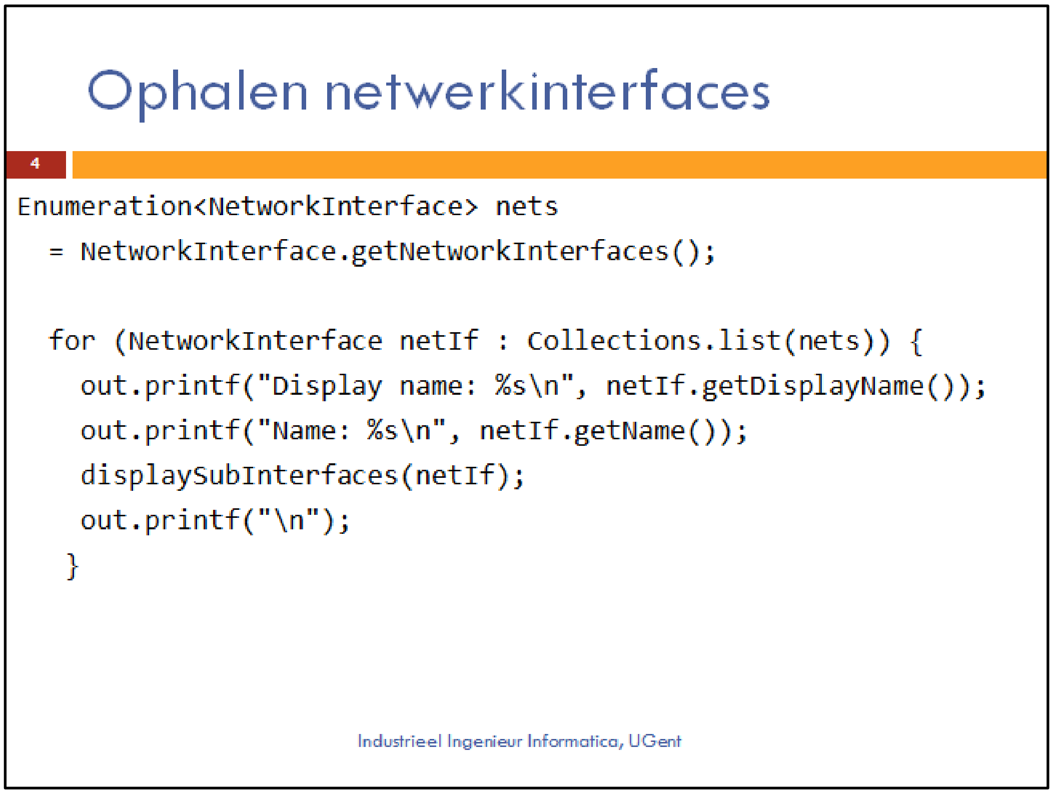
To send the data, the system determines which interface is used. However, if you have a preference or otherwise need to specify which NIC to use, you can query the system for the appropriate interfaces and find an address on the interface you want to use. When you create the socket and bind it to that address, the system uses the associated interface. For example:

```
NetworkInterface nif = NetworkInterface.getByName("bge0");
Enumeration<InetAddress> nifAddresses = nif.getInetAddresses();
Socket soc = new java.net.Socket();
soc.bind(new InetSocketAddress(nifAddresses.nextElement(), 0));
soc.connect(new InetSocketAddress(address, port));
```

You can also use NetworkInterface to identify the local interface on which a multicast group is to be joined. For example:

```
NetworkInterface nif = NetworkInterface.getByName("bge0");
MulticastSocket ms = new MulticastSocket();
ms.joinGroup(new InetSocketAddress(hostname, port), nif);
```

NetworkInterface can be used with Java APIs in many other ways beyond the two uses described here.



Ophalen netwerkinterfaces

4

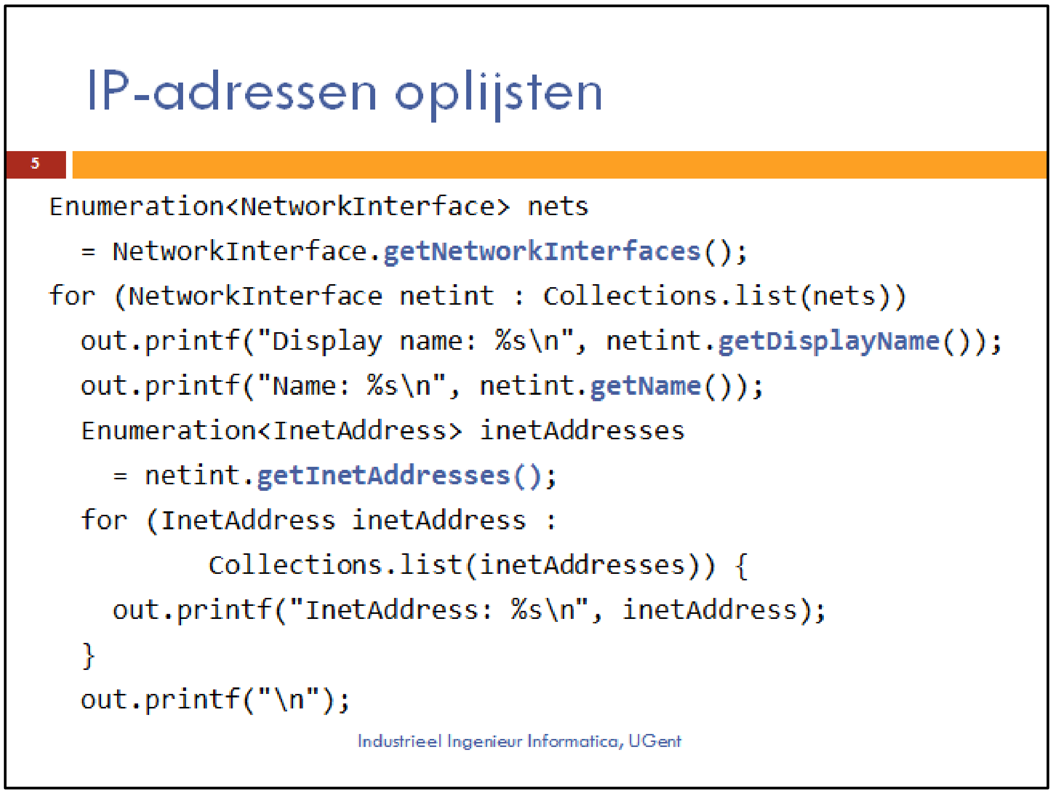
```
Enumeration<NetworkInterface> nets
  = NetworkInterface.getNetworkInterfaces();

  for (NetworkInterface netIf : Collections.list(nets)) {
    out.printf("Display name: %s\n", netIf.getDisplayName());
    out.printf("Name: %s\n", netIf.getName());
    displaySubInterfaces(netIf);
    out.printf("\n");
   }
```

Industrieel Ingenieur Informatica, UGent

The NetworkInterface class has no public constructor. Therefore, you cannot just create a new instance of this class with the new operator. Instead, the following static methods are available so that you can retrieve the interface details from the system:
getByInetAddress(),
getByName(),
and getNetworkInterfaces().

The first two methods are used when you already know the IP address or the name of the particular interface. The third method, getNetworkInterfaces() returns the complete list of interfaces on the machine.

Network interfaces can be hierarchically organized. The NetworkInterface class includes two methods, getParent() and getSubInterfaces(), that are pertinent to a network interface hierarchy. The getParent() method returns the parent NetworkInterface of an interface. If a network interface is a subinterface, getParent() returns a non-null value. The getSubInterfaces() method returns all the subinterfaces of a network interface.



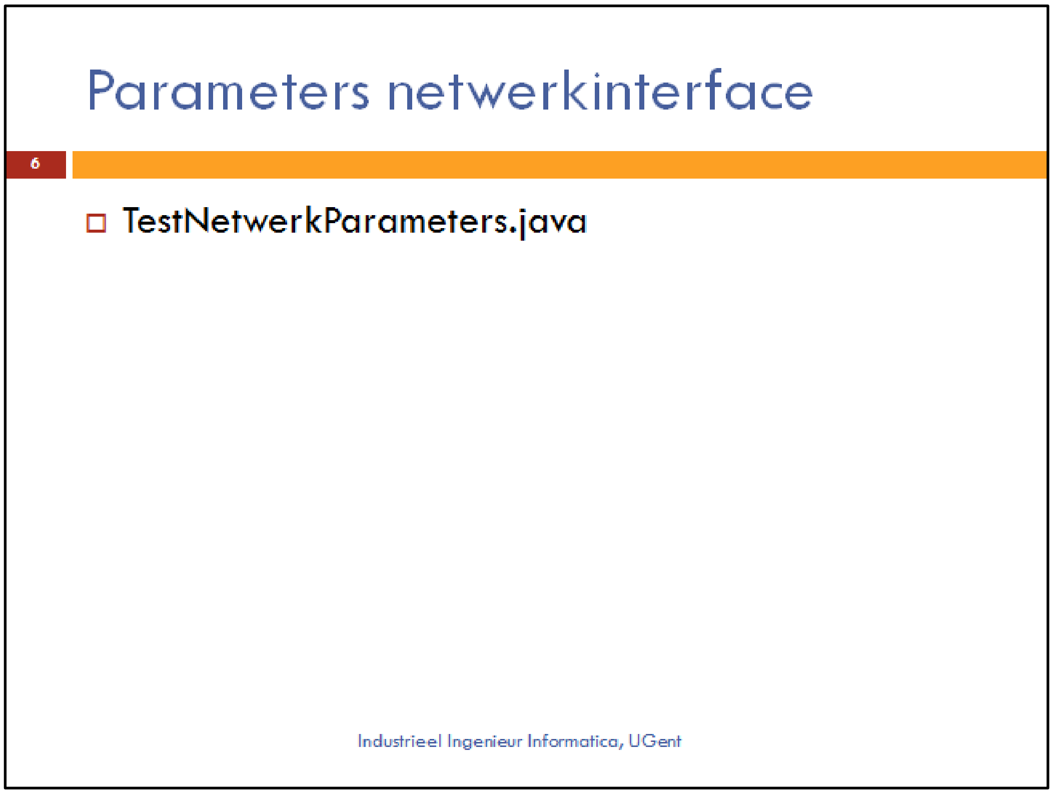
IP-adressen oplijsten

5

```
Enumeration<NetworkInterface> nets
  = NetworkInterface.getNetworkInterfaces();
for (NetworkInterface netint : Collections.list(nets))
  out.printf("Display name: %s\n", netint.getDisplayName());
  out.printf("Name: %s\n", netint.getName());
  Enumeration<InetAddress> inetAddresses
    = netint.getInetAddresses();
  for (InetAddress inetAddress :
          Collections.list(inetAddresses)) {
    out.printf("InetAddress: %s\n", inetAddress);
  }
  out.printf("\n");
```

Industrieel Ingenieur Informatica, UGent

One of the most useful pieces of information you can get from a network interface is the list of IP addresses that are assigned to it. You can obtain this information from a NetworkInterface instance by using one of two methods. The first method, getInetAddresses(), returns an Enumeration of InetAddress. The other method, getInterfaceAddresses(), returns a list of java.net.InterfaceAddress instances. This method is used when you need more information about an interface address beyond its IP address. For example, you might need additional information about the subnet mask and broadcast address when the address is an IPv4 address, and a network prefix length in the case of an IPv6 address.



Parameters netwerkinterface

6

- TestNetwerkParameters.java

Industrieel Ingenieur Informatica, UGent

You can access network parameters about a network interface beyond the name and IP addresses assigned to it.
You can discover if a network interface is "up" (that is, running) with the isUP() method. The following methods indicate the network interface type:

isLoopback() indicates if the network interface is a loopback interface.
isPointToPoint() indicates if the interface is a point-to-point interface.
isVirtual() indicates if the interface is a virtual interface.

The supportsMulticast() method indicates whether the network interface supports multicasting. The getHardwareAddress() method returns the network interface's physical hardware address, usually called MAC address, when it is available. The getMTU() method returns the Maximum Transmission Unit (MTU), which is the largest packet size.