

UNIT 2

Internet Addresses

(4 Hours) — Network Programming (CAC355)

Disclaimer: These notes were generated automatically from the CACS355 syllabus and may contain errors or omissions. Please verify important details (syntax, method names, default values) against your textbook (Java Network Programming, E. R. Harold) and class notes before the exam.

2.1 The InetAddress Class

Every host on an IP network is identified by an **IP address**. In Java, the `java.net.InetAddress` class represents an IP address together with (optionally) the host name. It has two subclasses, `Inet4Address` and `Inet6Address`. It has no public constructors; objects are created by static factory methods.

Creating new InetAddress objects

Method	Purpose
<code>getByName(String host)</code>	Returns the address of a host name or dotted/IPv6 literal (uses DNS if needed)
<code>getAllByName(String host)</code>	Returns an array of all addresses of the host (e.g., load-balanced sites)
<code>getByAddress(byte[] addr)</code>	Creates an address from raw bytes without a DNS lookup
<code>getByAddress(String host, byte[] addr)</code>	Creates an address with a given host name and raw bytes
<code>getLocalHost()</code>	Returns the address of the local machine
<code>getLoopbackAddress()</code>	Returns the loopback address (127.0.0.1 or ::1)

```
import java.net.*;
public class AddrDemo {
    public static void main(String[] args) throws Exception {
        InetAddress a = InetAddress.getByName("www.example.com");
        System.out.println(a); // www.example.com/93.184.216.34
        InetAddress[] all = InetAddress.getAllByName("www.example.com");
        InetAddress me = InetAddress.getLocalHost();
        System.out.println(me);
    }
}
```

Listing 2.1 — Creating `InetAddress` objects

*Note: These methods throw **UnknownHostException** (a checked exception) if the host cannot be resolved. Results may be cached by the JVM.*

Getter methods

Method	Returns
<code>getHostName()</code>	Host name; performs a reverse DNS lookup if the name is not known
<code>getCanonicalHostName()</code>	Fully qualified domain name (forces a reverse lookup)
<code>getHostAddress()</code>	IP address as a text string, e.g., "192.168.1.5"
<code>etAddress()</code>	IP address as a byte array (network byte order); bytes are signed in Java

Example — Reading address bytes

`etAddress()` returns signed bytes, so a value above 127 appears negative. Convert with **b & 0xFF** (or **b + 256** when negative) to print the usual dotted form.

```
byte[] b = InetAddress.getByName("192.168.1.5").getAddress();
for (byte x : b) System.out.print((x & 0xFF) + " ");    // 192 168 1 5
```

Address types

The `InetAddress` class provides methods to test what kind of address it holds:

Method	True when the address is...
<code>isAnyLocalAddress()</code>	the wildcard address (0.0.0.0 or ::)
<code>isLoopbackAddress()</code>	the loopback address (127.x.x.x or ::1)
<code>isLinkLocalAddress()</code>	link-local (169.254.x.x or fe80::/10)
<code>isSiteLocalAddress()</code>	private/site-local (10.x, 172.16–31.x, 192.168.x)
<code>isMulticastAddress()</code>	multicast (224.0.0.0–239.255.255.255, ff00::/8)
<code>isMCGlobal()</code> / <code>isMCOrgLocal()</code> / <code>isMCSiteLocal()</code> / <code>isMCLinkLocal()</code> / <code>isMCNodeLocal()</code>	a multicast address of the corresponding scope

Testing reachability

`isReachable(int timeout)` (and an overload that takes a `NetworkInterface`, TTL and timeout) tries to reach the host, using ICMP echo if permitted or a TCP connection to port 7 otherwise. It returns **true** if the host responded within the timeout in milliseconds. A false result does not always mean the host is down, since firewalls may block the probe.

```
InetAddress h = InetAddress.getByName("www.example.com");
System.out.println(h.isReachable(3000)); // wait up to 3 seconds
```

Object methods

- **equals(Object)** — two `InetAddress` objects are equal if their IP addresses are equal; the host name is ignored.
- **hashCode()** — computed from the IP address only.
- **toString()** — form “hostname/address”; the host name part is empty if unknown.

2.2 Inet4Address and Inet6Address

Inet4Address represents a 32-bit IPv4 address and **Inet6Address** a 128-bit IPv6 address. Usually you do not use these classes directly, because `InetAddress` returns the right subclass automatically and its methods work for both. Use `instanceof` when you need to know the version.

Feature	IPv4 (Inet4Address)	IPv6 (Inet6Address)
Size	32 bits (4 bytes)	128 bits (16 bytes)
Notation	Dotted decimal, 192.168.1.5	Hex groups, 2001:db8::1
<code>getAddress()</code> length	4	16
Extra methods	<code>isAnyLocalAddress</code> etc.	<code>isIPv4CompatibleAddress()</code> , <code>getScopeId()</code>

2.3 The NetworkInterface Class

`NetworkInterface` represents a local network interface — a physical NIC (e.g., `eth0`, `wlan0`) or a virtual one (e.g., `lo`). It is useful for finding the local addresses and for choosing which interface to use for sockets or multicast.

Factory methods

Method	Purpose
getByName(String name)	Interface by name, e.g., "eth0" (null if not found)
getByInetAddress(InetAddress a)	Interface that has this local address
getByIndex(int index)	Interface by index number
getNetworkInterfaces()	Enumeration of all interfaces on this machine

Getter methods

Method	Returns
getName() / getDisplayName()	System name / human-readable name
getInetAddresses()	Enumeration of addresses bound to this interface
getInterfaceAddresses()	List of InterfaceAddress (address + prefix length + broadcast)
getHardwareAddress()	MAC address as a byte array (may be null)
getMTU()	Maximum transmission unit
isUp(), isLoopback(), isVirtual(), isPointToPoint(), supportsMulticast()	Boolean status flags
getParent(), getSubInterfaces()	Virtual interface relationships

```
Enumeration<NetworkInterface> e = NetworkInterface.getNetworkInterfaces();
while (e.hasMoreElements()) {
    NetworkInterface ni = e.nextElement();
    System.out.println(ni.getName() + " up=" + ni.isUp());
    Enumeration<InetAddress> ips = ni.getInetAddresses();
    while (ips.hasMoreElements()) System.out.println("    " + ips.nextElement());
}
```

Listing 2.2 — Listing all interfaces and their addresses

2.4 Some Useful Programs

SpamCheck

Many spam-blacklist services (DNSBL) answer DNS queries. To check whether an IP such as **a.b.c.d** is listed, reverse its octets and append the blacklist domain (e.g., *d.c.b.a.sbl.spamhaus.org*). If the name resolves, the address is listed; if `UnknownHostException` occurs, it is not.

```

public static boolean isSpammer(String ip) {
    try {
        String[] q = ip.split("\\.");
        String name = q[3] + "." + q[2] + "." + q[1] + "." + q[0] + ".sbl.spamhaus.org";
        InetAddress.getBy_name(name);
        return true; // resolved -> listed
    } catch (UnknownHostException e) {
        return false; // not found -> not listed
    }
}

```

Listing 2.3 — Idea of the SpamCheck program

Processing web server logfiles

A web server log line begins with the client IP address, e.g. `192.168.1.5 - - [10/Jan/2025:10:15:32] "GET /index.html HTTP/1.1" 200 1024`. A program can read each line, take the text up to the first space, and convert it to a host name with `InetAddress.getBy_name(ip).getHostName()`. Because reverse DNS lookups are slow, real programs use **multithreading** (a thread pool) so many lookups run in parallel and preserve the output order.

```

BufferedReader in = new BufferedReader(new FileReader("access.log"));
String line;
while ((line = in.readLine()) != null) {
    int i = line.indexOf(' ');
    String ip = line.substring(0, i);
    String rest = line.substring(i);
    String host = InetAddress.getBy_name(ip).getHostName();
    System.out.println(host + rest);
}

```

Listing 2.4 — Replacing IPs with host names in a log file (sequential version)

Key Takeaways

- ✓ `InetAddress` has no public constructors; use `getByName`, `getAllByName`, `getByAddress`, `getLocalHost`.
- ✓ `getHostName` may trigger a reverse DNS lookup; `getHostAddress` returns the text IP; `getAddress` returns signed bytes.
- ✓ Type tests: `isLoopbackAddress`, `isSiteLocalAddress`, `isMulticastAddress`, etc.; `isReachable(timeout)` tests connectivity.
- ✓ `Inet4Address` (32-bit) and `Inet6Address` (128-bit) are the concrete subclasses.
- ✓ `NetworkInterface` gives access to local interfaces, addresses, MAC and MTU.
- ✓ Useful programs: DNSBL spam check and log-file reverse lookup (use threads for speed).

— *End of Unit 2* —

bcalink.com