Java Networking

Difficulty Level : Easy • Last Updated : 07 Sep, 2021

When computing devices such as laptops, desktops, servers, smartphones, and tablets and an eternally-expanding arrangement of IoT gadgets such as cameras, door locks, doorbells, refrigerators, audio/visual systems, thermostats, and various sensors are sharing information and data with each other is known as networking.

In simple words, the term network programming or networking associates with writing programs that can be executed over various computer devices, in which all the devices are connected to each other to share resources using a network. Here, we are going to discuss ***Java Networking***.

- What is Java Networking?
- Common Network Protocols
- Java Network Terminology
- Java Networking Classes
- Java Networking Interfaces
- Socket Programming
- Inet Address

Networking supplements a lot of power to simple programs. With networks, a single program can regain information stored in millions of computers positioned anywhere in the world. Java is the leading programming language composed from scratch with networking in mind. Java Networking is a notion of combining two or more computing devices together to share resources.

All the Java program communications over the network are done at the application layer. The **java.net** package of the J2SE APIs comprises various classes and interfaces that execute the low-level communication features, enabling the user to formulate programs that focus on resolving the problem.

Common Network Protocols

As stated earlier, the **java.net** package of the Java programming language includes various classes and interfaces that provide an easy-to-use means to access network resources. Other than classes and interfaces, the **java.net** package also provides support for the two well-known network protocols. These are:

1. **Transmission Control Protocol (TCP) –** TCP or Transmission Control Protocol allows secure communication between different applications. TCP is a connection-oriented protocol which means that once a connection is established, data can be transmitted in two directions. This protocol is typically used over the Internet Protocol. Therefore, TCP is also referred to as TCP/IP. TCP has built-in methods to examine for errors and ensure the delivery of data in the order it was sent, making it a complete protocol for transporting information like still images, data files, and web pages.

2. **User Datagram Protocol (UDP) –** UDP or User Datagram Protocol is a connection-less protocol that allows data packets to be transmitted between different applications. UDP is a simpler Internet protocol in which error-checking and recovery services are not required. In UDP, there is no overhead for opening a connection, maintaining a connection, or terminating a connection. In UDP, the data is continuously sent to the recipient, whether they receive it or not.

Java Networking Terminology

In Java Networking, many terminologies are used frequently. These widely used Java Networking Terminologies are given as follows:

1. **IP Address –** An IP address is a unique address that distinguishes a device on the internet or a local network. IP stands for "Internet Protocol." It comprises a set of rules governing the format of data sent via the internet or local network. IP Address is referred to as a logical address that can be modified. It is composed of octets. The range of each octet varies from 0 to 255.
   - Range of the IP Address – 0.0.0.0 to 255.255.255.255
   - For Example – 192.168.0.1

2. **Port Number –** A port number is a method to recognize a particular process connecting internet or other network information when it reaches a server. The port number is used to identify different applications uniquely. The port number behaves as a communication endpoint among applications. The port number is correlated with the IP address for transmission and communication among two applications. There are 65,535 port numbers, but not all are used every day.

3. **Protocol –** A network protocol is an organized set of commands that define how data is transmitted between different devices in the same network. Network protocols are the reason through which a user can easily communicate with people all over the world and thus play a critical role in modern digital communications. For Example – TCP, FTP, POP, etc.

4. **MAC Address –** MAC address stands for Media Access Control address. It is a bizarre identifier that is allocated to a NIC (Network Interface Controller/ Card). It contains a 48 bit or 64-bit address, which is combined with the network adapter. MAC address can be in hexadecimal composition. In simple words, a MAC address is a unique number that is used to track a device in a network.

number so that the TCP layer can recognize the application to which the data is intended to be sent.

6. **Connection-oriented and connection-less protocol –** In a connection-oriented service, the user must establish a connection before starting the communication. When the connection is established, the user can send the message or the information, and after this, they can release the connection. However, In connectionless protocol, the data is transported in one route from source to destination without verifying that the destination is still there or not or if it is ready to receive the message. Authentication is not needed in the connectionless protocol.
   - Example of Connection-oriented Protocol – Transmission Control Protocol (TCP)
   - Example of Connectionless Protocol – User Datagram Protocol (UDP)

Java Networking classes

The **java.net** package of the Java programming language includes various classes that provide an easy-to-use means to access network resources. The classes covered in the **java.net** package are given as follows –

1. **CacheRequest –** The CacheRequest class is used in java whenever there is a need to store resources in ResponseCache. The objects of this class provide an edge for the OutputStream object to store resource data into the cache.

2. **CookieHandler –** The CookieHandler class is used in Java to implement a callback mechanism for securing up an HTTP state management policy implementation inside the HTTP protocol handler. The HTTP state management mechanism specifies the mechanism of how to make HTTP requests and responses.

3. **CookieManager –** The CookieManager class is used to provide a precise implementation of CookieHandler. This class separates the storage of cookies from the policy surrounding accepting and rejecting cookies. A CookieManager

provides tools for the production of datagram packets for connectionless transmission applying the datagram socket class.

5. **InetAddress** – The InetAddress class is used to provide methods to get the IP address of any hostname. An IP address is expressed by a 32-bit or 128-bit unsigned number. InetAddress can handle both IPv4 and IPv6 addresses.

6. **Server Socket** – The ServerSocket class is used for implementing system-independent implementation of the server-side of a client/server Socket Connection. The constructor for ServerSocket class throws an exception if it can't listen on the specified port. For example – it will throw an exception if the port is already being used.

7. **Socket** – The Socket class is used to create socket objects that help the users in implementing all fundamental socket operations. The users can implement various networking actions such as sending, reading data, and closing connections. Each Socket object built using **java.net.Socket** class has been connected exactly with 1 remote host; for connecting to another host, a user must create a new socket object.

8. **DatagramSocket** – The DatagramSocket class is a network socket that provides a connection-less point for sending and receiving packets. Every packet sent from a datagram socket is individually routed and delivered. It can further be practiced for transmitting and accepting broadcast information. Datagram Sockets is Java's mechanism for providing network communication via UDP instead of TCP.

9. **Proxy** – A proxy is a changeless object and a kind of tool or method or program or system, which serves to preserve the data of its users and computers. It behaves like a wall between computers and internet users. A Proxy Object represents the Proxy settings to be applied with a connection.

. **URL** – The URL class in Java is the entry point to any available sources on the internet. A Class URL describes a Uniform Resource Locator, which is a signal to a "resource" on the World Wide Web. A source can denote a simple file or directory, a connection of a resource as defined by a similar URL. The URLConnection class is used for assisting two distinct yet interrelated purposes. Firstly it provides control on interaction with a server(especially an HTTP server) than a URL class. Furthermore, with a URLConnection, a user can verify the header transferred by the server and can react consequently. A user can also configure header fields used in client requests using URLConnection.

Java Networking Interfaces

The **java.net** package of the Java programming language includes various interfaces also that provide an easy-to-use means to access network resources. The interfaces included in the **java.net** package are as follows:

1. **CookiePolicy –** The CookiePolicy interface in the **java.net** package provides the classes for implementing various networking applications. It decides which cookies should be accepted and which should be rejected. In CookiePolicy, there are three pre-defined policy implementations, namely ACCEPT_ALL, ACCEPT_NONE, and ACCEPT_ORIGINAL_SERVER.

2. **CookieStore –** A CookieStore is an interface that describes a storage space for cookies. CookieManager combines the cookies to the CookieStore for each HTTP response and recovers cookies from the CookieStore for each HTTP request.

3. **FileNameMap –** The FileNameMap interface is an uncomplicated interface that implements a tool to outline a file name and a MIME type string. FileNameMap charges a filename map (known as a mimetable) from a data file.

4. **SocketOption –** The SocketOption interface helps the users to control the behavior of sockets. Often, it is essential to develop necessary features in Sockets. SocketOptions allows the user to set various standard options.

5. **SocketImplFactory –** The SocketImplFactory interface defines a factory for SocketImpl instances. It is used by the socket class to create socket

the name of the protocol family.

Socket Programming

Java Socket programming is practiced for communication between the applications working on different JRE. Sockets implement the communication tool between two computers using TCP. Java Socket programming can either be connection-oriented or connection-less. In Socket Programming, Socket and ServerSocket classes are managed for connection-oriented socket programming. However, DatagramSocket and DatagramPacket classes are utilized for connection-less socket programming.

A client application generates a socket on its end of the communication and strives to combine that socket with a server. When the connection is established, the server generates an object of socket class on its communication end. The client and the server can now communicate by writing to and reading from the socket.

The **java.net.Socket** class describes a socket, and the **java.net.ServerSocket** class implements a tool for the server program to host clients and build connections with them.

Steps to establishing a TCP connection between two computing devices using Socket Programming

The following are the steps that occur on establishing a TCP connection between two computers using socket programming are given as follows:

Step 1 – The server instantiates a ServerSocket object, indicating at which port number communication will occur.

Step 2 – After instantiating the ServerSocket object, the server requests the accept() method of the ServerSocket class. This program pauses until a client connects to the server on the given port.

Step 3 – After the server is idling, a client instantiates an object of Socket class, defining the server name and the port number to connect to.

Step 5 – On the server-side, the accept() method returns a reference to a new socket on the server connected to the client's socket.

After the connections are stabilized, communication can happen using I/O streams. Each object of a socket class has both an OutputStream and an InputStream. The client's OutputStream is correlated to the server's InputStream, and the client's InputStream is combined with the server's OutputStream. Transmission Control Protocol (TCP) is a two-way communication protocol. Hence information can be transmitted over both streams at the corresponding time.

Socket Class

The **Socket class** is used to create socket objects that help the users in implementing all fundamental socket operations. The users can implement various networking actions such as sending, reading data, and closing connections. Each Socket object created using **java.net.Socket** class has been correlated specifically with 1 remote host. If a user wants to connect to another host, then he must build a new socket object.

Methods of Socket Class

In Socket programming, both the client and the server have a Socket object, so all the methods under the Socket class can be invoked by both the client and the server. There are many methods in the Socket class.

S No.	Method	Description
1	**public void connect(SocketAddress host, int timeout)**	This method is used to connect the socket to the particularized host. This method is required only when the user instantiates the Socket applying the no-argument constructor.

		which the socket is pinned on the remote machine.
3	**public InetAddress getInetAddress()**	This method is used to return the location of the other computer to which the socket is connected.
4	**public int getLocalPort()**	This method is used to return the port to which the socket is joined on the local machine.
5	**public SocketAddress getRemoteSocketAddress()**	This method returns the location of the remote socket.
6	**public InputStream getInputStream()**	This method is used to return the input stream of the socket. This input stream is combined with the output stream of the remote socket.
7	**public OutputStream getOutputStream()**	This method is used to return the output stream of the socket. The output stream is combined with the input stream of the remote socket.
8	**public void close()**	This method is used to close the socket, which causes the object of the Socket class to no longer be able to connect again to any server.

ServerSocket Class

The **ServerSocket class** is used for providing system-independent implementation of the server-side of a client/server Socket Connection. The constructor for

There are many methods in the ServerSocket class which are very useful for the users. These methods are:

S no.	Method	Description
1	**public int getLocalPort()**	This method is used to return the port that the server socket is monitoring on. This method is beneficial if a user passed 0 as the port number in a constructor and lets the server find a port for him.
2	**public void setSoTimeout(int timeout)**	This method is used to set the time-out value for the time in which the server socket pauses for a client during the accept() method.
3	**public Socket accept()**	This method waits for an incoming client. This method is blocked till either a client combines to the server on the specified port or the socket times out, considering that the time-out value has been set using the setSoTimeout() method. Otherwise, this method will be blocked indefinitely.
4	**public void bind(SocketAddress host, int backlog)**	This method is used to bind the socket to the particularized server and port in the object of SocketAddress. The user should use this method if he has instantiated the ServerSocket using the no-argument constructor.

ample of Socket Programming in Java:

The below example illustrates a pretty basic one-way Client and Server setup where a Client connects, sends messages to the server and the server shows them using a

```java
// A Java program for a ClientSide

import java.io.*;
import java.net.*;

public class clientSide {

    // initialize socket and input output streams
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;

    // constructor to put ip address and port
    public clientSide(String address, int port)
    {

        // establish a connection
        try {

            socket = new Socket(address, port);

            System.out.println("Connected");

            // takes input from terminal
            input = new DataInputStream(System.in);

            // sends output to the socket
            out = new DataOutputStream(
                socket.getOutputStream());
        }

        catch (UnknownHostException u) {

            System.out.println(u);
        }

        catch (IOException i) {

            System.out.println(i);
        }

        // string to read message from input
        String line = "";

        // keep reading until "End" is input
```

```
                out.writeUTF(line);
            }

            catch (IOException i) {

                System.out.println(i);
            }
        }

        // close the connection
        try {

            input.close();

            out.close();

            socket.close();
        }

        catch (IOException i) {

            System.out.println(i);
        }
    }

    public static void main(String[] args)
    {

        clientSide client
            = new clientSide("127.0.0.1", 5000);
    }
}
```

Server Side Java Implementation:

```
// A Java program for a serverSide
import java.io.*;
import java.net.*;

public class serverSide {

    // initialize socket and input stream
    {

        // starts server and waits for a connection
        try {
            server = new ServerSocket(port);

            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

            socket = server.accept();

            System.out.println("Client accepted");

            // takes input from the client socket
            in = new DataInputStream(
                new BufferedInputStream(
                    socket.getInputStream()));

            String line = "";

            // reads message from client until "End" is sent
            while (!line.equals("End")) {

                try {

                    line = in.readUTF();

                    System.out.println(line);
                }

                catch (IOException i) {

                    System.out.println(i);
                }
            }

            System.out.println("Closing connection");

            // close connection
            socket.close();

            in.close();
        }

        catch (IOException i) {
```

```
        serverSide server = new serverSide(5000);
    }
}
```

To run on Terminal or Command Prompt

Open two windows one for Server and another for Client.

1. First run the Server application. It will show –

```
Server started
Waiting for a client …
```

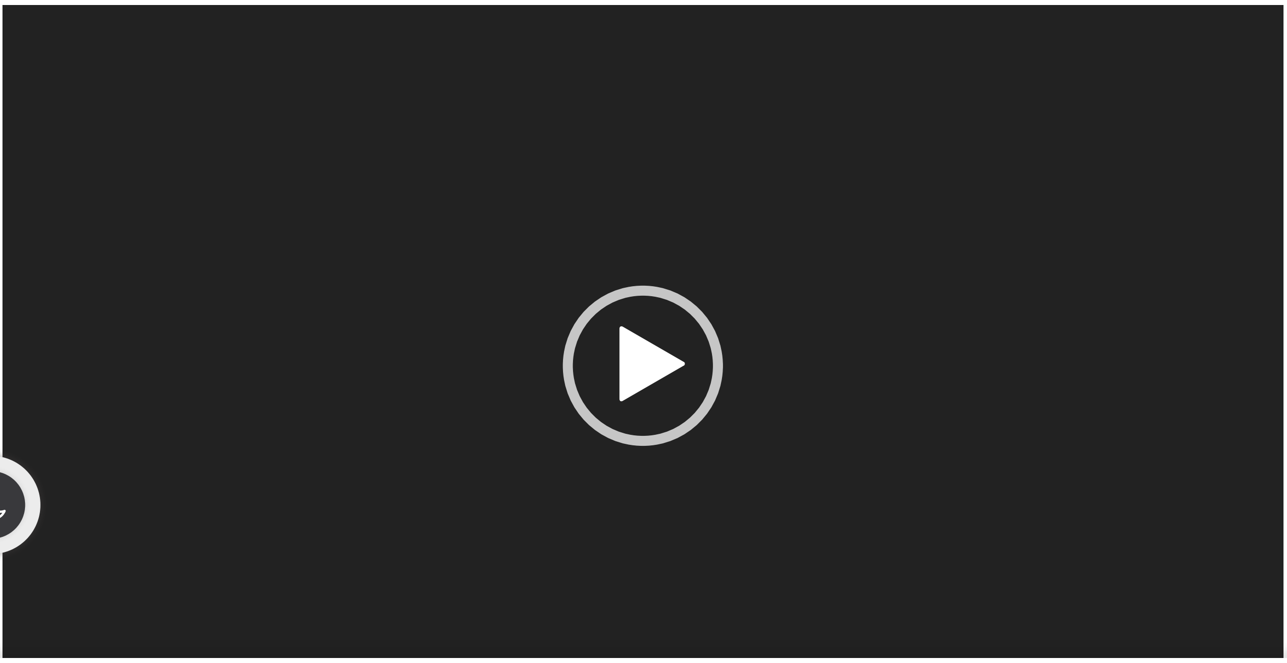
2. Then run the Client application on another terminal. It will show:

```
Connected
```

and the server accepts the client and shows,

```
Client accepted
```

3. Then you can start typing messages in the Client window. Here is the sample video of the output.



Methods of InetAddress Class

Java InetAddress class represents an IP address. The following given are the important methods of the InetAddress class –

S No.	Method	Description
1	**static InetAddress getByAddress(byte[] addr)**	This method is used to return an object of the InetAddress class provided the raw IP address.
2	**static InetAddress getByAddress(String host, byte[] addr)**	This method is used to create an InetAddress based on the given hostname and IP address.
3	**static InetAddress getByName(String host)**	This method is used to determine the IP address of a host when the host's name is given.
4	**static InetAddress InetAddress getLocalHost()**	This method is used to return the localhost.

		address.
6	**String getHostAddress()**	This method returns the IP address in the form of a string in a textual display.
7	**String toString()**	This method is used to convert the IP address to a string.

Examples of Inet Address Class Methods:
The Java implementation of the **Inet Address** class to illustrate the usage of methods is shown below:

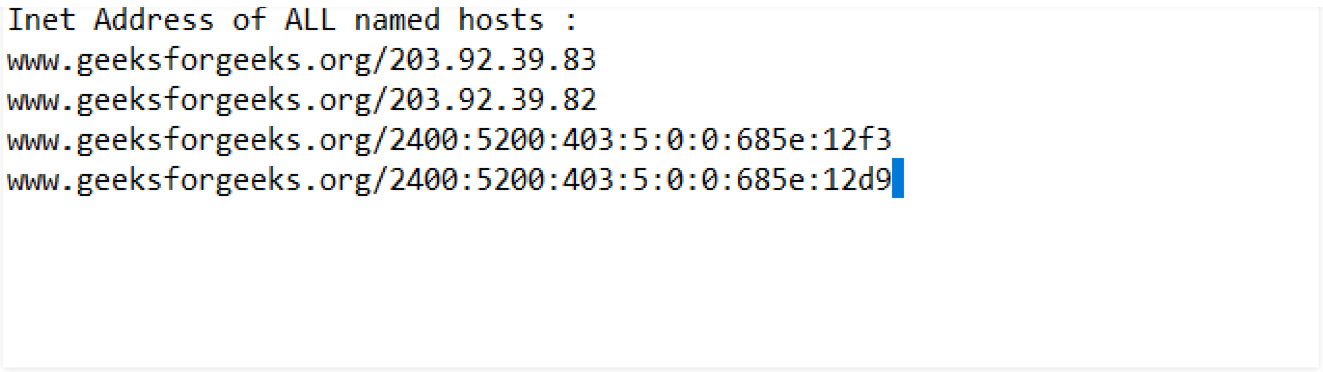
Example 1:

```
import java.net.*;

public class InetAddressExample1 {

    public static void main(String[] args) throws UnknownHostException{


          // To get and print InetAddress of the Local Host
        InetAddress address = InetAddress.getLocalHost();

        System.out.println("InetAddress of the Local Host : "+address);

        // To get and print host name of the Local Host
        String hostName=address.getHostName();

        System.out.println("\nHost name of the Local Host : "+hostName);

    }
```

Example 2:

```
import java.net.*;

public class InetAddressExample2 {

    public static void main(String[] args)
        throws UnknownHostException
    {

        // To get and print InetAddress of Named Hosts
        InetAddress address1 = InetAddress.getByName(
                                   "write.geeksforgeeks.org");

        System.out.println("Inet Address of named hosts : "
                                            + address1);

        // To get and print ALL InetAddress of Named Host
        InetAddress arr[] = InetAddress.getAllByName(
                                    "www.geeksforgeeks.org");

        System.out.println("\nInet Address of ALL named hosts :");

        for (int i = 0; i < arr.length; i++) {

            System.out.println(arr[i]);
        }
    }
}
```

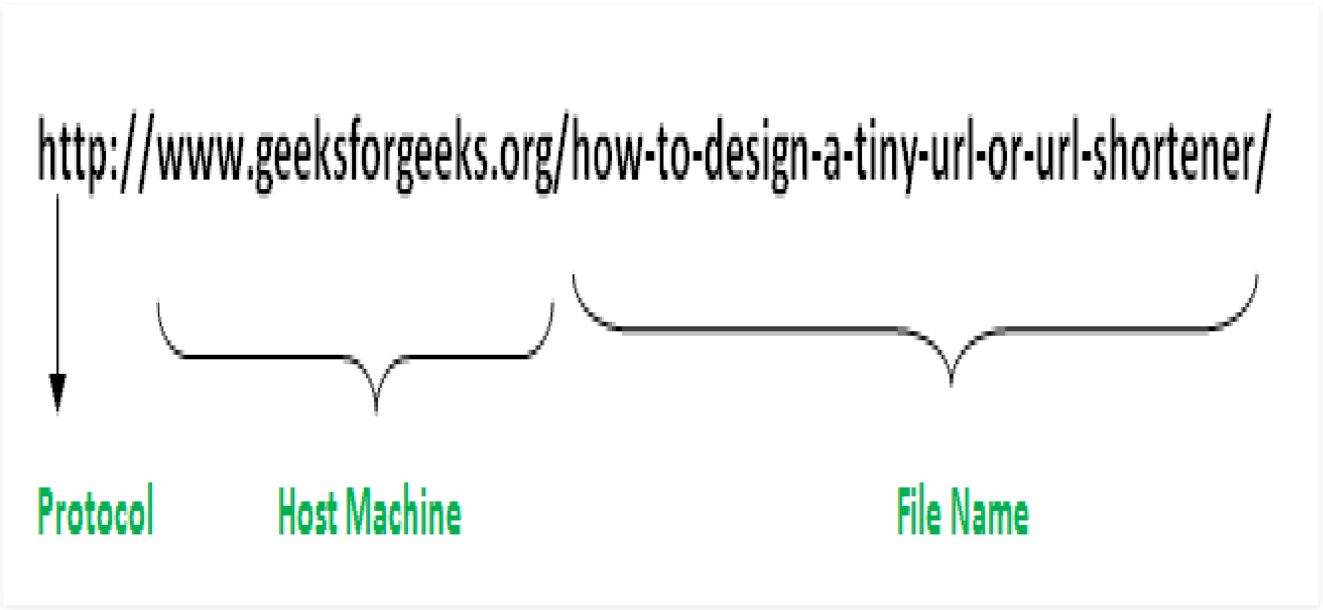
Output



```
Inet Address of ALL named hosts :
www.geeksforgeeks.org/203.92.39.83
www.geeksforgeeks.org/203.92.39.82
www.geeksforgeeks.org/2400:5200:403:5:0:0:685e:12f3
www.geeksforgeeks.org/2400:5200:403:5:0:0:685e:12d9
```

URL Class

The URL class in Java is the entry point to any available sources on the internet. A Class URL describes a Uniform Resource Locator, which is a signal to a "resource" on the World Wide Web. A source can denote a simple file or directory, or it can indicate a more difficult object, such as a query to a database or a search engine. URL is a string of text that recognizes all the sources on the Internet, showing us the address of the source, how to interact with it, and recover something from it.


http://www.geeksforgeeks.org/how-to-design-a-tiny-url-or-url-shortener/
Protocol
Host Machine
File Name

Components of a URL

URL can have many forms. The most general however follows a three-components system-

4. **Port Number –** The port number is used to identify different applications uniquely. It is typically optional.

Methods of Java URL Class

There are many methods in Java URL Class that are commonly used in Java Networking. These methods are:

S. No.	Methods	Description
1	**public String getProtocol()**	This method returns the protocol that is used by the URL.
2	**public String getHost()**	This method returns the hostname of the URL in IPv6 composition.
3	**public int getPort()**	This method returns the port associated with the protocol specified by the URL.
4	**public String getFile()**	This method returns the filename.
5	**public String getPath()**	This method returns the path of the URL, or null if empty.
6	**public String toString()**	This method is used to return the string representation of the provided URL object.
7	**public int getDefaultPort()**	This method returns the default port used.

Examples of URL Class Methods

```
import java.net.*;

public class URLclassExample1 {

    public static void main(String[] args)
        throws MalformedURLException
    {

        // creates a URL with string representation.
        URL url = new URL(
            "https://write.geeksforgeeks.org/post/3038131");

        // print the string representation of the URL
        String s = url.toString();

        System.out.println("URL :" + s);
    }
}
```

Output

```
URL :https://write.geeksforgeeks.org/post/3038131
```

Example 2:

```
import java.net.*;

public class URLclassExample2 {

    public static void main(String[] args)
        throws MalformedURLException
    {

        URL url = new URL(
            "https://write.geeksforgeeks.org/post/3038131");

        // to get and print the protocol of the URL
        String protocol = url.getProtocol();
        System.out.println("HostName : " + host);

        // to get and print the file name of the URL
        String fileName = url.getFile();

        System.out.println("File Name : " + fileName);
    }
}
```

Output

```
Protocol : https
HostName : write.geeksforgeeks.org
File Name : /post/3038131
```

Example 3:

```
import java.net.*;

public class URLclassExample3 {

    public static void main(String[] args)
        throws MalformedURLException
    {

        URL url = new URL(
            "https://write.geeksforgeeks.org/post/3038131");

        // to get and print the default port of the URL
        int defaultPort = url.getDefaultPort();

        System.out.println("Default Port : " + defaultPort);

        // to get and print the path of the URL
        String path = url.getPath();

        System.out.println("Path : " + path);
    }
}
```

This was a brief introduction to Java Networking. In this article, many important topics like Introduction of Java Networking, Common Network Protocols, Java Network Terminology, Java Networking Classes, Java Networking Interfaces, Socket Programming, Inet Address, and URL Class were covered.

Previous Next

RECOMMENDED ARTICLES

Page : 1 2 3

01 **Networking in Java | Set 1 (Java.net.InetAddress class)**
18, May 16

2 **Android Networking Bare Skin - Understanding JPost**
15, Aug 21

05 **Different Ways to Convert java.util.Date to java.time.LocalDate in Java**
05, Jan 21

06 **How to Convert java.util.Date to java.sql.Date in Java?**

04 How to Convert java.sql.Date to java.util.Date in Java?
02, Feb 21

08 Java.Lang.Float class in Java
27, Mar 17

Article Contributed By :

nishkarshgandhi
@nishkarshgandhi

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Article Tags : Java

Practice Tags : Java

Improve Article Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Start Your Coding Journey Now! Login Register

Company

About Us

Careers

Privacy Policy

Contact Us

Copyright Policy

Learn

Algorithms

Data Structures

Languages

CS Subjects

Video Tutorials

Web Development

Web Tutorials

HTML

CSS

JavaScript

Bootstrap

Contribute

Write an Article

Write Interview Experience

Internships

Videos