

Java Programming

Module Name: **Programming with Java & HTML**

Module Code: **CST 06102**

Instructors: **Jamila L.**

Contents

1. Explain Java programming environment
2. Explain how to install Java programming environment
3. Use Java programming environment to create simple Java programs
4. Write a program to prompt the user for a hostname and then looks up the IP address for the hostname and displays the results.
5. Write a program to read the webpage from a website and display the contents of the webpage.

Java programming environment: JDK

The Java Development Kit (JDK) is a software development environment that offers a collection of tools and libraries necessary for developing Java applications. You need the JDK to convert your source code into a format that the Java Runtime Environment (JRE) can execute.

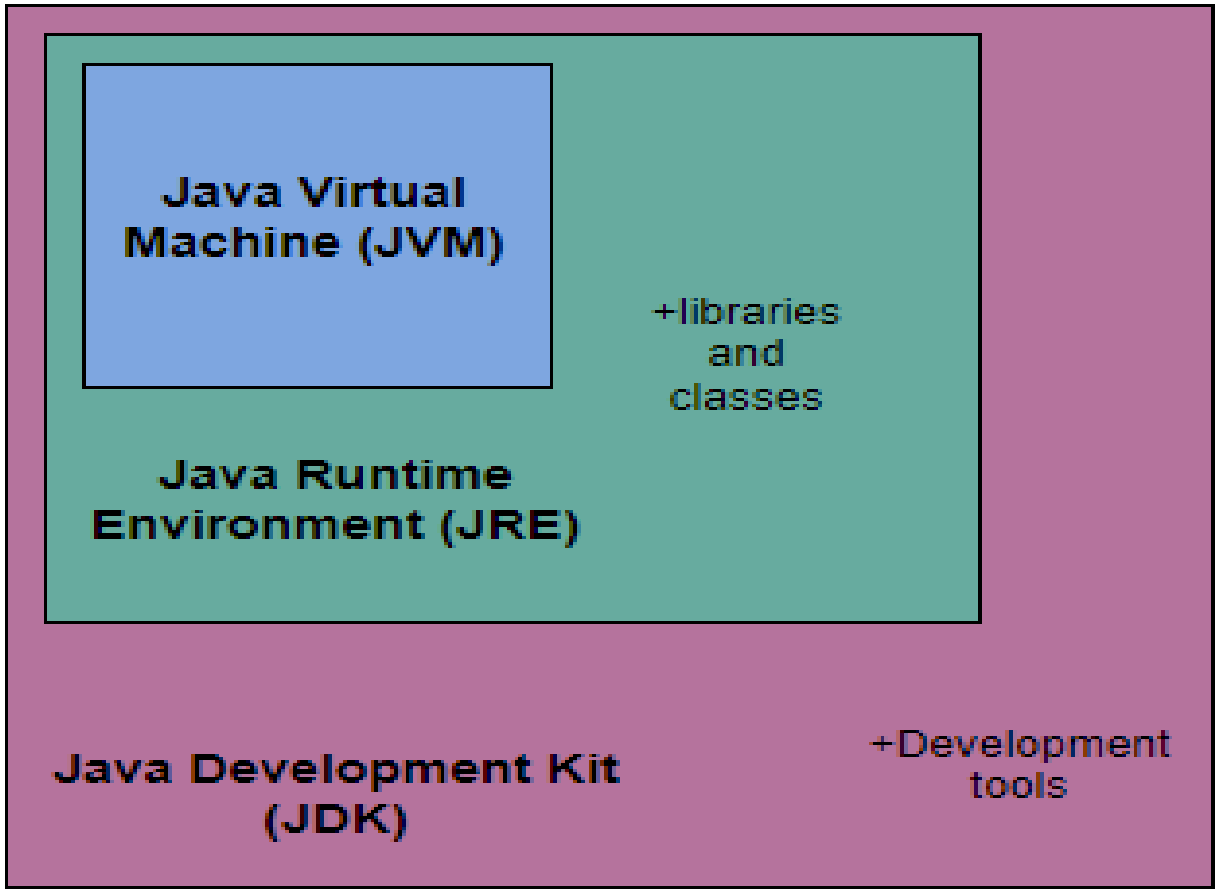
The JDK includes the Java Runtime Environment (JRE), an interpreter (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc), and some other development tools.

The Java Runtime Environment itself consists of the Java Virtual Machine (JVM), supporting files, and core classes.

Java programming environment: JDK

Typically, if you are only interested in running Java programs on your machine or browser, you only need to install JRE.

However, if you would like to develop an application and do Java programming, you will need JDK.



What is the JVM?

- The Java Virtual Machine is a program which provides the runtime environment to execute Java programs. Java programs cannot run if a supporting JVM is not available.
- On running a Java program from the command line like —

```
java <arguments> <program name>
```

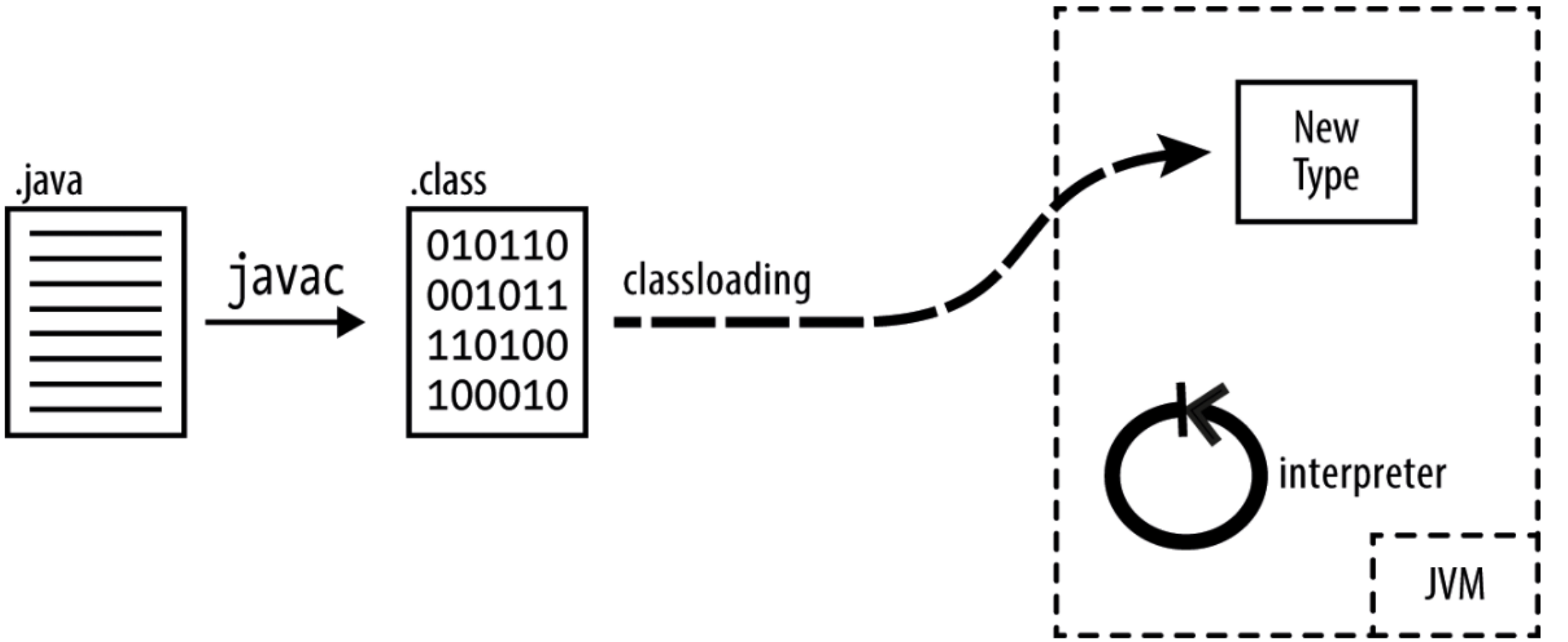
- the OS would bring up the JVM as a process and then execute the program in the freshly started (and empty) Virtual Machine.

JVM cont...

- JVM does not take Java source files as input. The java source code is first converted to bytecode by javac program. Javac takes in source files as input and outputs bytecode in the form of class files with .class extension.

- These class files are then interpreted (stepped through one at a time) by the JVM interpreter and the program is executed.

Java file compilation



What is Java IDE?

Java IDE (Integrated Development Environment) is a software application that enables users to write and debug Java programs more easily.

We will use VS Code IDE

In order to run Java within Visual Studio Code, you need to install a JDK. The Extension Pack for Java supports Java version 1.5 or above.

Java Networking

What Is Java Networking?

Java networking is the concept of connecting two or more computing devices to share resources.

The application layer of a Java program communicates with the network. The java.net package contains all of the Java networking classes and interfaces.

Java Networking Terminologies

The widely used Java networking terminologies are given below:

- IP Address
- Protocol
- Port Number
- MAC Address
- Connection-oriented and connection-less protocol
- Socket

Network Protocols

The java.net package supports 2 protocols. These are their names:

1. **The Transmission Control Protocol (TCP):**

TCP enables secure communication between two applications. It is commonly applied together with the Internet Protocol, which is referred to as TCP/IP.

2. **User Datagram Protocol (UDP):**

UDP is a connectionless protocol allowing data packets to be sent between applications.

java.net package

The java.net package can be divided into two sections:

- **A Low-Level API:** It deals with the abstractions of addresses i.e. networking identifiers, Sockets i.e. bidirectional data communication mechanism and Interfaces i.e. network interfaces.
- **A High Level API:** It deals with the abstraction of URIs i.e. Universal Resource Identifier, URLs i.e. Universal Resource Locator, and Connections i.e. connections to the resource pointed by URLs.
- **The java.net package provides many classes to deal with networking applications in Java.** A list of these classes is given below:

Java Networking Classes

The Java programming language's java.net package contains a number of classes that make it simple to access network resources. The following classes are included in the java.net package:

- HttpRequest, HttpClient,HttpResponse
- CookieHandler
- CookieManager
- DatagramPacket
- InetAddress
- ServerSocket
- Socket
- Proxy
- URL & URLConnection etc…

Java Socket

In Java, a socket is one end of a two-way communication link between two programs running on a network.

- A socket is always associated with a port number to allow the TCP layer to identify the application to which data is being sent.
- The Socket class is used to create socket objects that aid users in performing all basic socket operations. Users can perform various networking tasks such as sending data, reading data, and closing connections.

Socket Programming

Socket programming is a method of communicating between two nodes on a network.

One socket (node) listens on a specific port at an IP address, while the other socket reaches out to form a connection. While the client attempts to contact the server, the server creates the listener socket.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

Socket Programming cont…

The steps involved in establishing a TCP connection between two computers using socket programming are as follows:

Step 1 - The server creates a ServerSocket object and specifies which port number communication will take place.

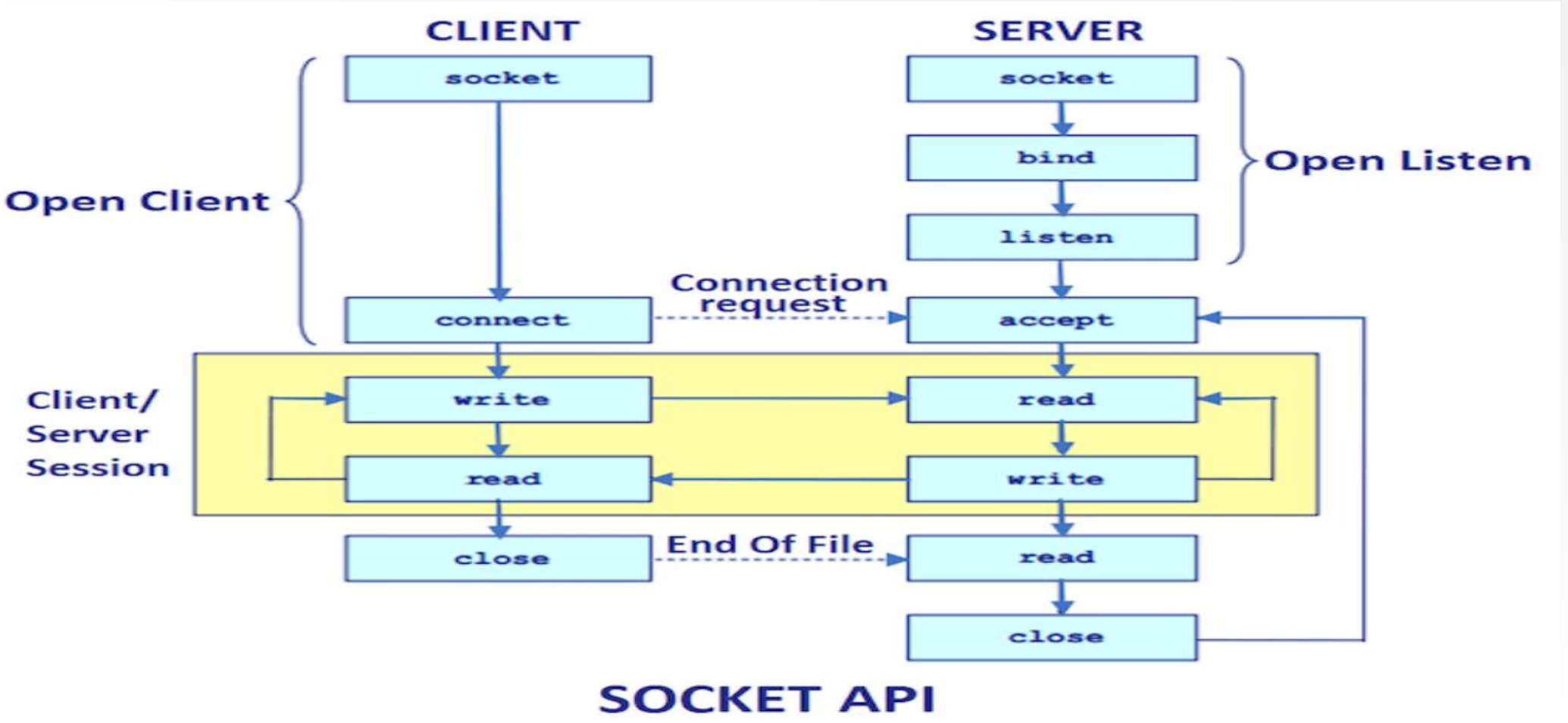
Step 2 - After the ServerSocket object is instantiated, the server invokes the ServerSocket class's accept() method. This program waits for a client to connect to the server on the specified port.

Step 3 - While the server is idle, a client creates an object of the Socket class and specifies the server name and port number to connect to.

Step 4 - Following the preceding step, the function Object() { [native code] } of the Socket class attempts to connect the client to the specified server and port number. If communication is authenticated, the client receives a Socket object capable of interacting with the server.

Step 5 - The accept() method on the server returns a reference to a new socket on the server that is connected to the client's socket.

Socket API


CLIENT
SERVER
socket
socket
bind
listen
Open Client
Open Listen
Connection request
connect
accept
Client/ Server Session
write
read
read
write
close
End Of File
read
close
SOCKET API

Socket Programming cont…

After the connections have been stabilized, communication can take place via I/O streams.

- A socket class object has an OutputStream as well as an InputStream.
- The OutputStream of the client is connected to the server's InputStream, which is then combined with the server's OutputStream.
- TCP or Transmission Control Protocol is a two-way communication protocol. As a result, information can be transmitted over both streams simultaneously.

Example of Java Socket Programming

Creating Server:

To create the server application, we need to create the instance of ServerSocket class.

Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number.

The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

```
ServerSocket ss=new);  ServerSocket(6666

Socket s=ss.accept();//establishes connection and waits for the
client
```

Example of Java Socket Programming

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

Example: File: MyClient.java

```
import java.io.*;
import java.net.*;
public class MyClient {
 public static void main(String[] args) {
    try{
    Socket s=new Socket("localhost",6666);
    DataOutputStream dout=new DataOutputStream(s.getOutputStream());
    dout.writeUTF("Hello Server");
    dout.flush();
    dout.close();
    s.close();
    }catch(Exception e){System.out.println(e);}
 }
}
```

Example: File: MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer {
    public static void main(String[] args){
        try{
            ServerSocket ss=new ServerSocket(6666);
            Socket s=ss.accept();//establishes connection
            DataInputStream dis=new
DataInputStream(s.getInputStream());
            String  str=(String)dis.readUTF();
            System.out.println("message= "+str);
            ss.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

Java URL

The Java URL class represents an URL. URL is an acronym for Uniform Resource Locator. It points to a resource on the World Wide Web.

A URL contains many information:

- Protocol: In this case, http is the protocol.
- Server name or IP Address: In this case, www.javatpoint.com is the server name.
- Port Number: It is an optional attribute. If we write http//www.sjuit.ac.tz:80/index/ , 80 is the port number. If port number is not mentioned in the URL, it returns -1.
- File Name or directory name: In this case, index.jsp is the file name.

The java.net.URL class represents a URL and has a complete set of methods to manipulate URL in Java.

URL Constructors

Sr.No.	Constructors & Description
1	**public URL(String protocol, String host, int port, String file) throws MalformedURLException** Creates a URL by putting together the given parts.
2	**public URL(String protocol, String host, String file) throws MalformedURLException** Identical to the previous constructor, except that the default port for the given protocol is used.
3	**public URL(String url) throws MalformedURLException** Creates a URL from the given String.
4	**public URL(URL context, String url) throws MalformedURLException** Creates a URL by parsing together the URL and String arguments.

URL Example

```
import java.net.*;
public class URLDemo{
public static void main(String[] args){
try{
URL url=new URL("http://www.sjuit.ac.tz:57/index");
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());
}catch(Exception e){System.out.println(e);}
}
}
```

Java URLConnection Class

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.

Features of URLConnection class

- URLConnection is an abstract class. The two subclasses HttpURLConnection and JarURLConnection makes the connetion 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.

Java InetAddress class

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.alibaba.com, www.google.com, www.facebook.com, etc.
- An IP address is represented by 32-bit or 128-bit unsigned number.
- An instance of InetAddress represents the IP address with its corresponding host name.
- Moreover, InetAddress has a cache mechanism to store successful and unsuccessful host name resolutions.

Java InetAddress class

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.

Example of Java InetAddress class

```
import java.net.*;
public class ExapleInet{
  public static void main(String[] args){
    try{
      InetAddress my_address = InetAddress.getLocalHost();
      System.out.println("The IP address is: " + my_address.getHostAddress());
      System.out.println("The host name is: " + my_address.getHostName());
     }
     catch (UnknownHostException e){
        System.out.println( "Couldn't find the local address.");
      }
    }
  }
```

Java reading web page tools

Java has built-in tools and third-party libraries for reading/downloading web pages. The following are some of the methods to read a web page content:

- HttpClient,
- URL,
- JSoup,
- HtmlCleaner,
- Apache HttpClient,
- Jetty HttpClient, and
- HtmlUnit..

Java HttpClient Library

You can read the contents of a web page in several ways using Java.

Http client is a transfer library, it resides on the client side, sends and receives HTTP messages. It provides up to date, feature-rich and, efficient implementation which meets the recent HTTP standards.

- The GET request (of Http protocol) is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
- The HttpClient API provides a class named HttpGet which represents the get request method.

Java HttpClient

- We use the Java HttpClient to download the web page.

```
HttpClient client = HttpClient.newHttpClient();
```

- A new HttpClient is created with the newHttpClient factory method.

```
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create("http://webcode.me"))
        .build();
```

- We build a synchronous request to the webpage.

```
HttpResponse<String> response = client.send(request,
    HttpResponse.BodyHandlers.ofString());
System.out.println(response.body());
```

Java HttpClient Example

```
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
public class ContentReader2 {
    public static void main(String[] args) throws IOException, InterruptedException {
        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("https://zetcode.com/java/readwebpage/"))
                .GET() // GET is default
                .build();
        HttpResponse<String> response = client.send(request,
                HttpResponse.BodyHandlers.ofString());
        System.out.println(response.body());
    }
}
```

Thank you!!