دانشگاه شهید بهشتی

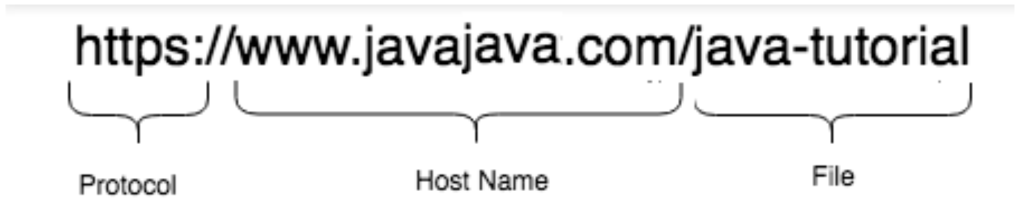
دانشکده مهندسی و علوم کامپیوتر

برنامه نویسی پیشرفته

وحیدی اصل

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. For example:

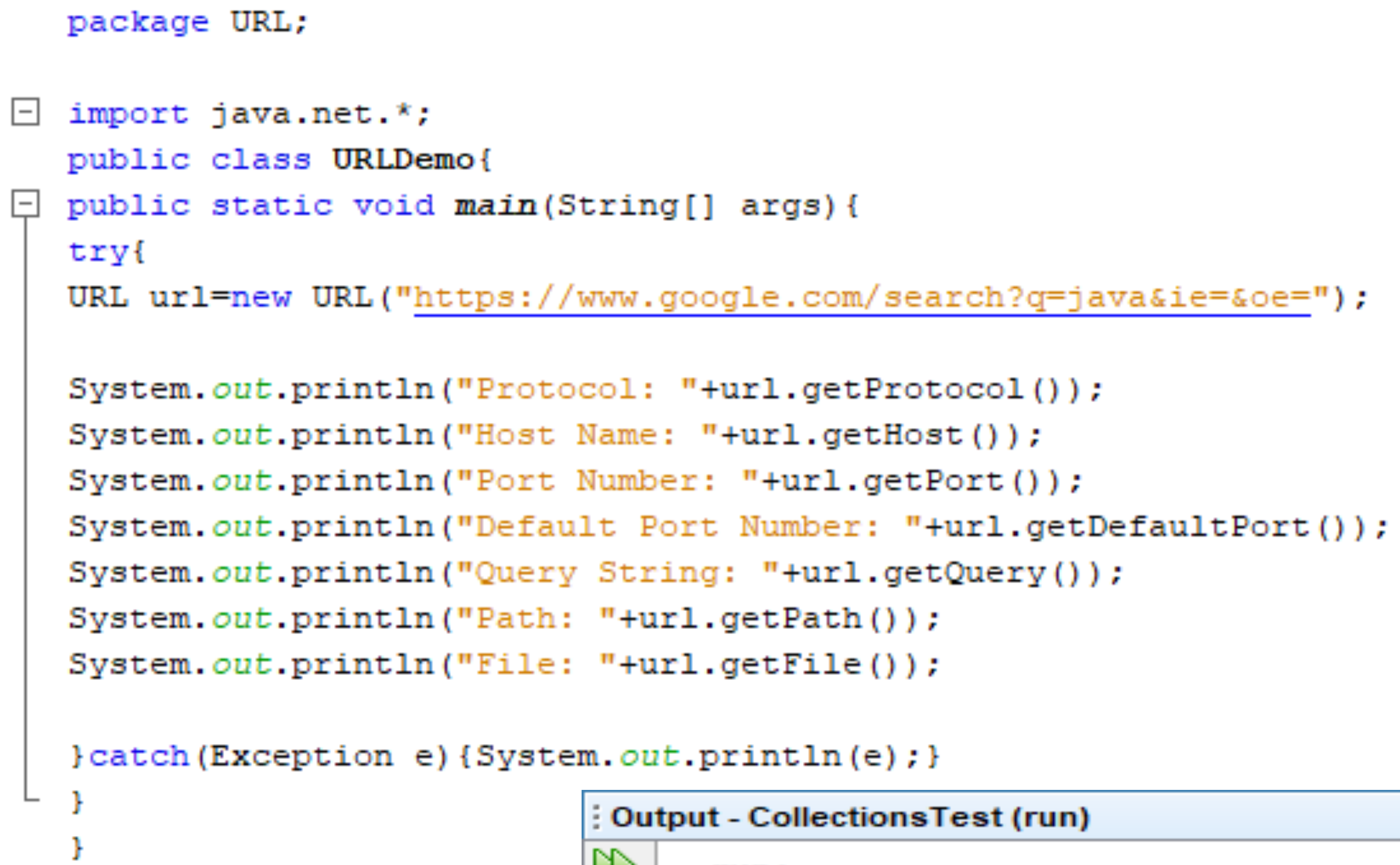


- A URL contains many information:
  - Protocol: In this case, http is the protocol.
  - Server name or IP Address: In this case, www.javajava.com is the server name.
- **Port Number:** It is an optional attribute. If we write http//ww.javajava.com:80/john/ , 80 is the port number. If port number is not mentioned in the URL, it returns -1.
- **File Name or directory name:** In this case, java-tutorial is the file name

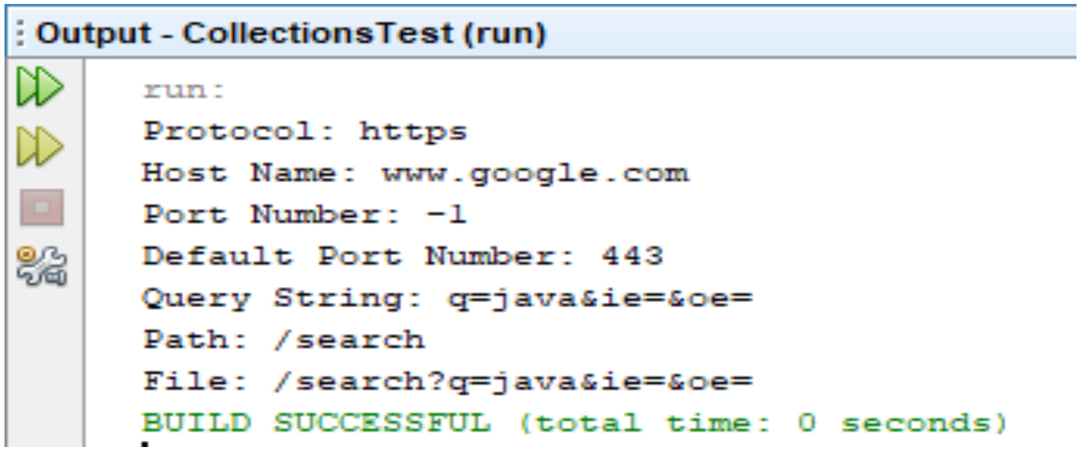
Commonly used methods of Java URL class

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 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.

Example of Java URL class



```java
package URL;

import java.net.*;
public class URLDemo{
public static void main(String[] args){
try{
URL url=new URL("https://www.google.com/search?q=java&ie=&oe=");

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());

}catch(Exception e){System.out.println(e);}
}
}
```



Output - CollectionsTest (run)

```
run:
Protocol: https
Host Name: www.google.com
Port Number: -1
Default Port Number: 443
Query String: q=java&ie=&oe=
Path: /search
File: /search?q=java&ie=&oe=
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java URLConnection class

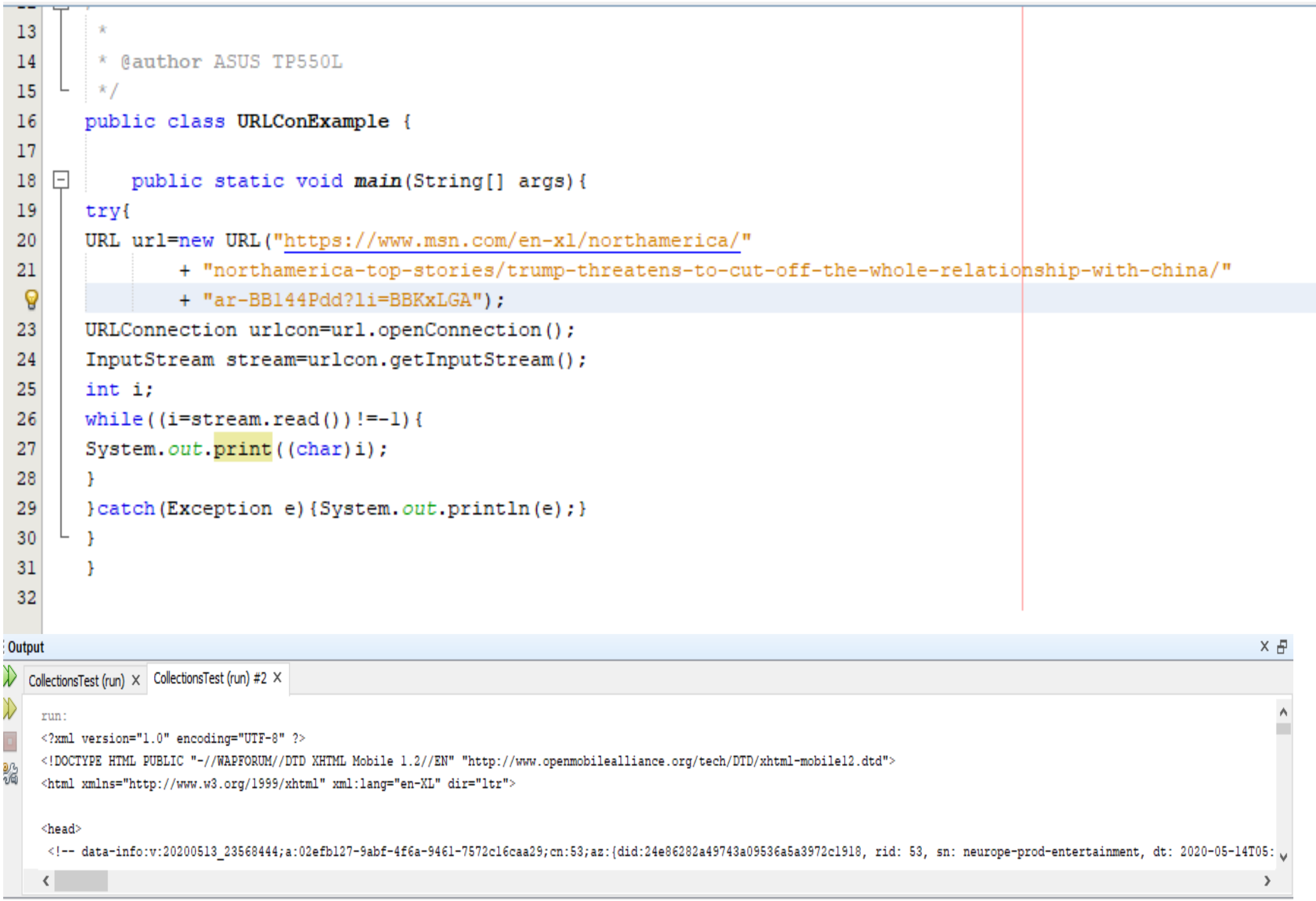
- The **Java URLConnection** class represents a communication link between the URL and the application.
- This class can be used to read and write data to the specified resource referred by the URL.

- How to get the object of URLConnection class
  - The openConnection() method of URL class returns the object of URLConnection class. Syntax:

```
public URLConnection openConnection()throws IOException{}
```

- Displaying source code of a webpage by URLConnecton class
  - The URLConnection class provides many methods, we can display all the data of a webpage by using the getInputStream() method.
  - The getInputStream() method returns all the data of the specified URL in the stream that can be read and displayed.

Java URLConnection class Example



```java
 *
 * @author ASUS TP550L
 */
public class URLConExample {

    public static void main(String[] args){
try{
URL url=new URL("https://www.msn.com/en-xl/northamerica/"
        + "northamerica-top-stories/trump-threatens-to-cut-off-the-whole-relationship-with-china/"
        + "ar-BB144Pdd?li=BBKxLGA");
URLConnection urlcon=url.openConnection();
InputStream stream=urlcon.getInputStream();
int i;
while((i=stream.read())!=-1){
System.out.print((char)i);
}
}catch(Exception e){System.out.println(e);}
}
}
```

Output

CollectionsTest (run) X | CollectionsTest (run) #2 X

```
run:
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE HTML PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.2//EN" "http://www.openmobilealliance.org/tech/DTD/xhtml-mobile12.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-XL" dir="ltr">

<head>
 <!-- data-info:v:20200513_23568444;a:02efb127-9abf-4f6a-9461-7572c16caa29;cn:53;az:{did:24e86282a49743a09536a5a3972c1918, rid: 53, sn: neurope-prod-entertainment, dt: 2020-05-14T05:
```

Java HttpURLConnection class

- The **Java HttpURLConnection** class is specific URLConnection. It works for HTTP protocol only.
  - By the help of HttpURLConnection class, you can get information of any HTTP URL such as header information, status code, response code etc.
  - The java.net.HttpURLConnection is subclass of URLConnection class.
- How to get the object of HttpURLConnection class
  - The openConnection() method of URL class returns the object of URLConnection class. Syntax:

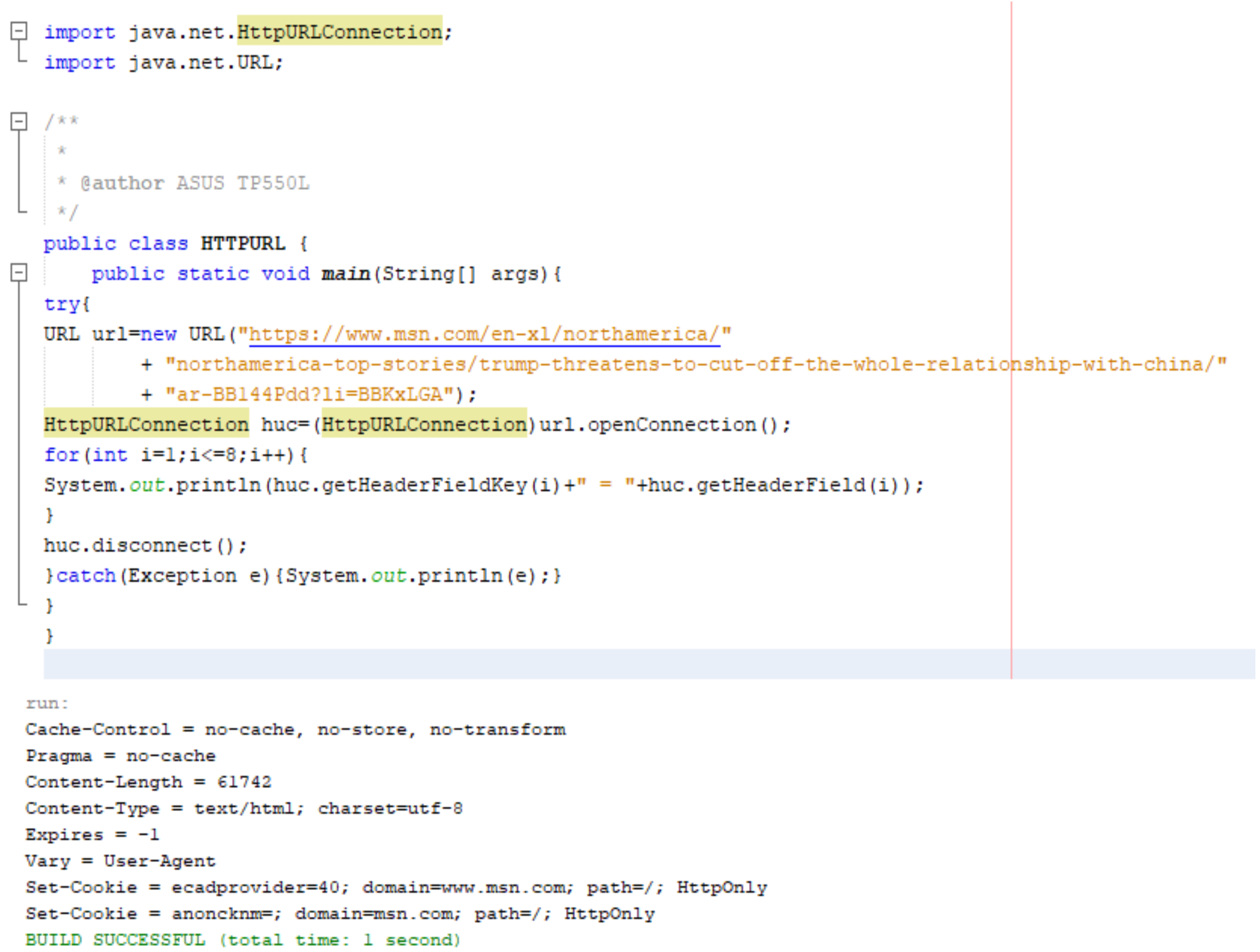
public URLConnection openConnection()**throws** IOException{}

- You can typecast it to HttpURLConnection type as given below.

URL url=**new** URL("http://www.java.com/java-tutorial");

HttpURLConnection huc=(HttpURLConnection)url.openConnection();

HTTPURLConnection class example



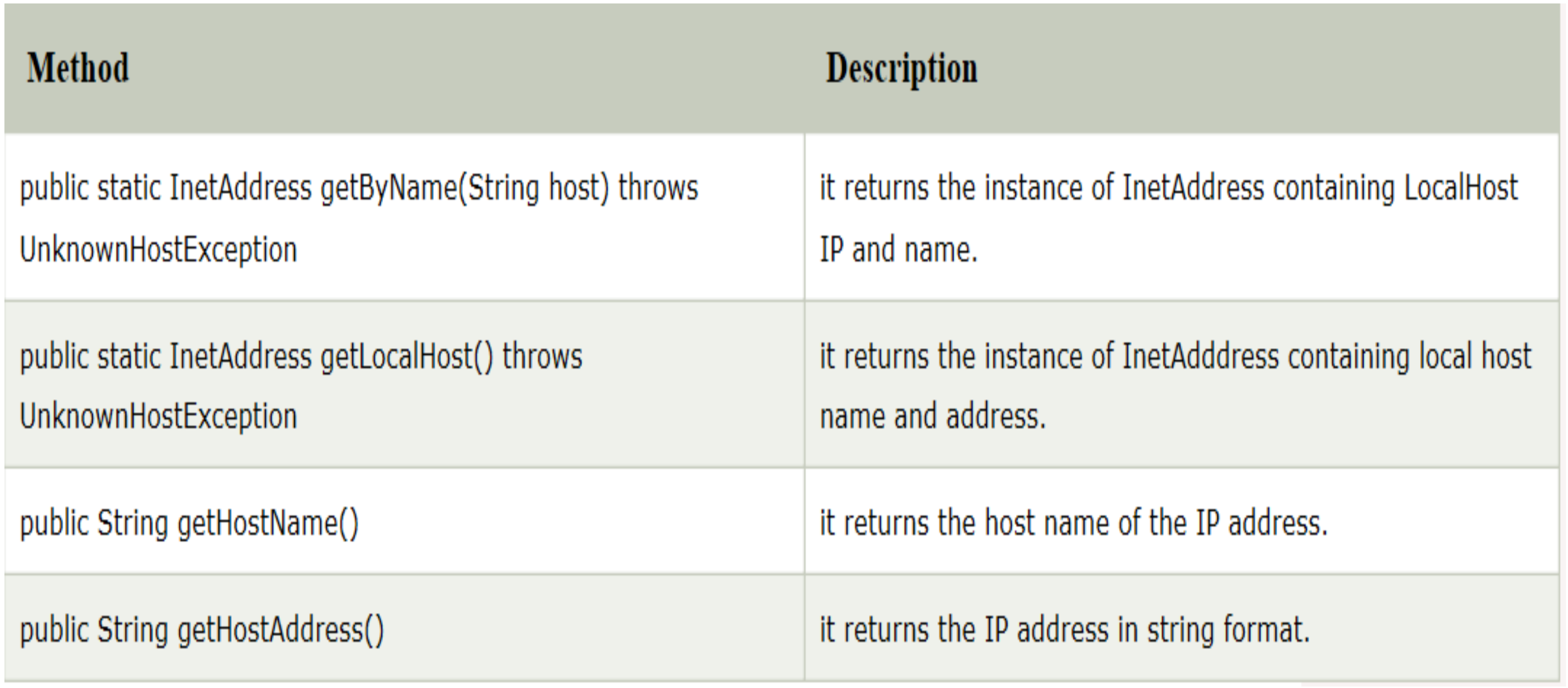
```java
import java.net.HttpURLConnection;
import java.net.URL;

/**
 *
 * @author ASUS TP550L
 */
public class HTTPURL {
    public static void main(String[] args){
try{
URL url=new URL("https://www.msn.com/en-xl/northamerica/"
        + "northamerica-top-stories/trump-threatens-to-cut-off-the-whole-relationship-with-china/"
        + "ar-BB144Pdd?li=BBKxLGA");
HttpURLConnection huc=(HttpURLConnection)url.openConnection();
for(int i=1;i<=8;i++){
System.out.println(huc.getHeaderFieldKey(i)+" = "+huc.getHeaderField(i));
}
huc.disconnect();
}catch(Exception e){System.out.println(e);}
}
}
```

```
run:
Cache-Control = no-cache, no-store, no-transform
Pragma = no-cache
Content-Length = 61742
Content-Type = text/html; charset=utf-8
Expires = -1
Vary = User-Agent
Set-Cookie = ecadprovider=40; domain=www.msn.com; path=/; HttpOnly
Set-Cookie = anoncknm=; domain=msn.com; path=/; HttpOnly
BUILD SUCCESSFUL (total time: 1 second)
```

Java InetAddress class

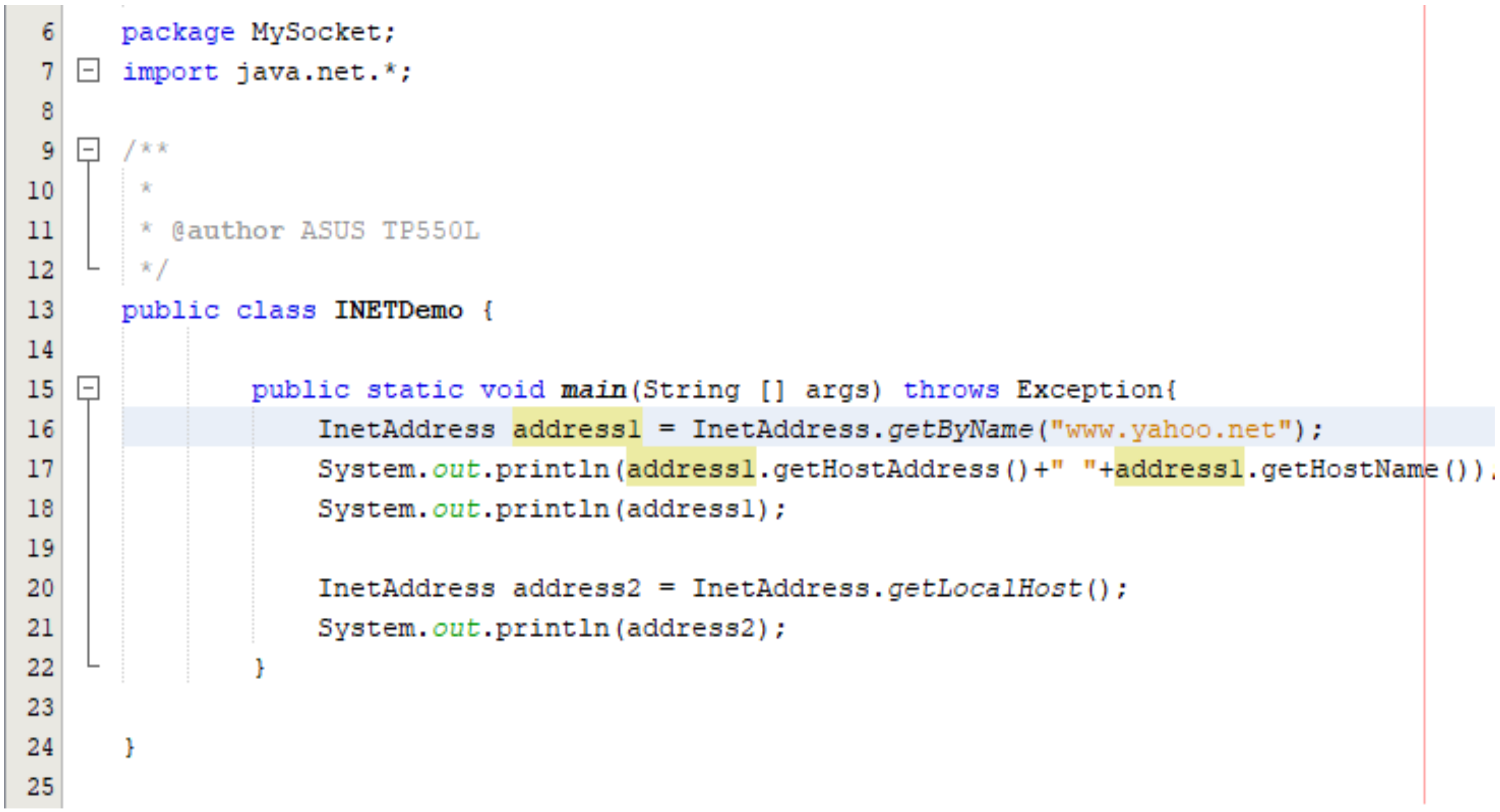
- **Java InetAddress** class represents an IP address.
- The java.net.InetAddress class provides methods to get the IP of any host name *for example* www.yahoo.com, www.google.com, www.facebook.com, etc.
- An IP address is represented by 32-bit or 128-bit unsigned number. An instance of InetAddress represents the IP address with its corresponding host name.
- Basically you create instances of this class to use with other classes: Socket, ServerSocket, DatagramPacket and Datagra mSocket.
- The InetAddress class doesn't have public constructors, so you create a new instance by using one of its factory methods shown in next slide.

Java InetAddress class methods

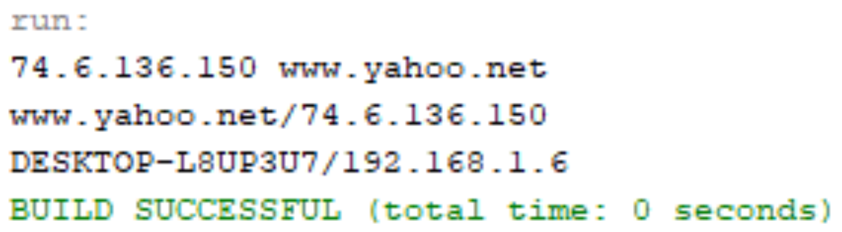


Method	Description
public static InetAddress getByName(String host) throws UnknownHostException	it returns the instance of InetAddress containing LocalHost IP and name.
public static InetAddress getLocalHost() throws UnknownHostException	it returns the instance of InetAdddress containing local host name 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.

مثال کلاس InetAdress



```
package MySocket;
import java.net.*;

/**
 *
 * @author ASUS TP550L
 */
public class INETDemo {

        public static void main(String [] args) throws Exception{
            InetAddress address1 = InetAddress.getByName("www.yahoo.net");
            System.out.println(address1.getHostAddress()+" "+address1.getHostName()),
            System.out.println(address1);

            InetAddress address2 = InetAddress.getLocalHost();
            System.out.println(address2);
        }

}
```



```
run:
74.6.136.150 www.yahoo.net
www.yahoo.net/74.6.136.150
DESKTOP-L8UP3U7/192.168.1.6
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java DatagramSocket and DatagramPacket

- Java DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

Java DatagramSocket

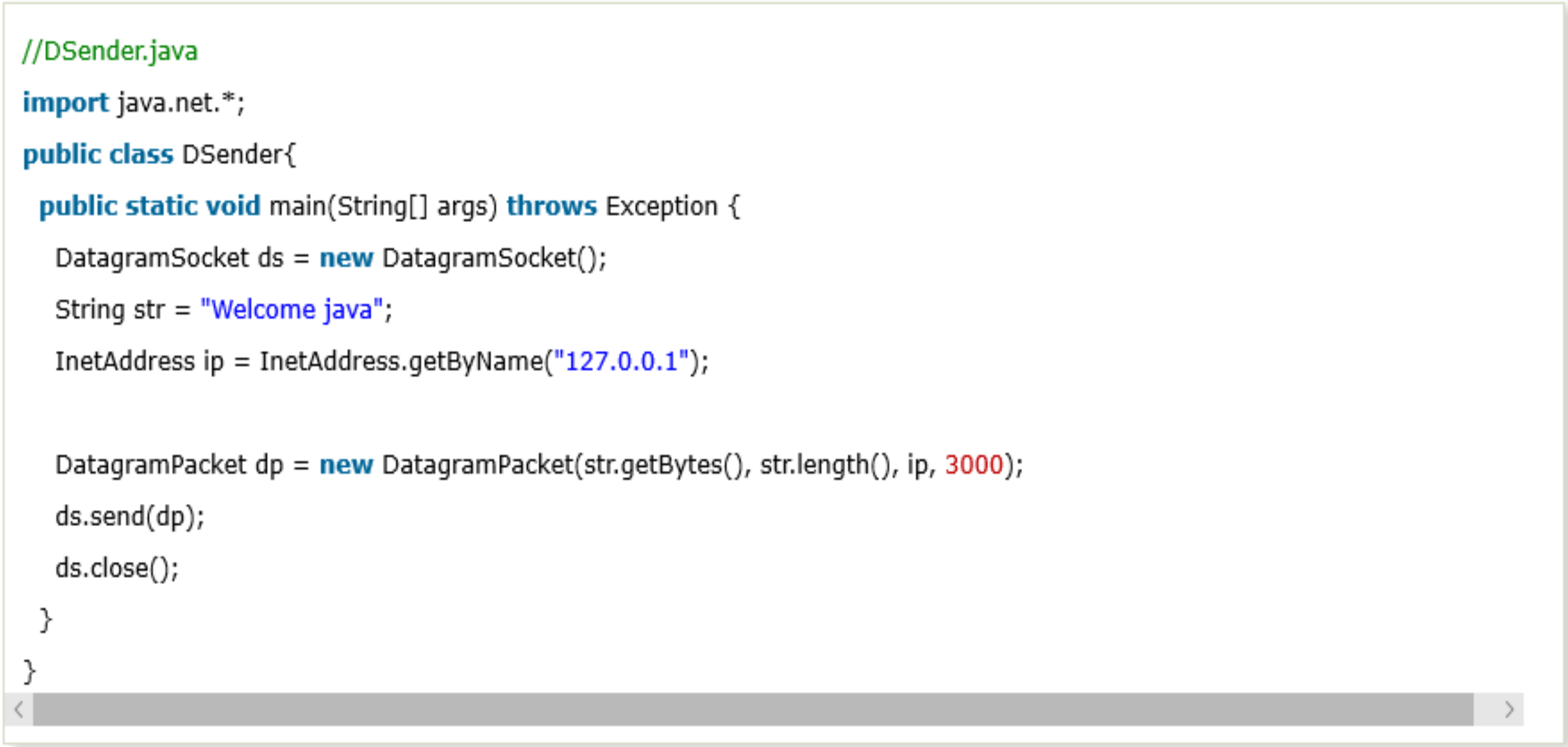
- Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.
- A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.
- Commonly used Constructors of DatagramSocket class
  - DatagramSocket() throws SocketException: it creates a datagram socket and binds it with the available Port Number on the localhost machine.
  - DatagramSocket(int port) throws SocketException: it creates a datagram socket and binds it with the given Port Number.
  - DatagramSocket(int port, InetAddress address) throws SocketException: it creates a datagram socket and binds it with the specified port number and host address.

Java DatagramPacket class

- Java DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

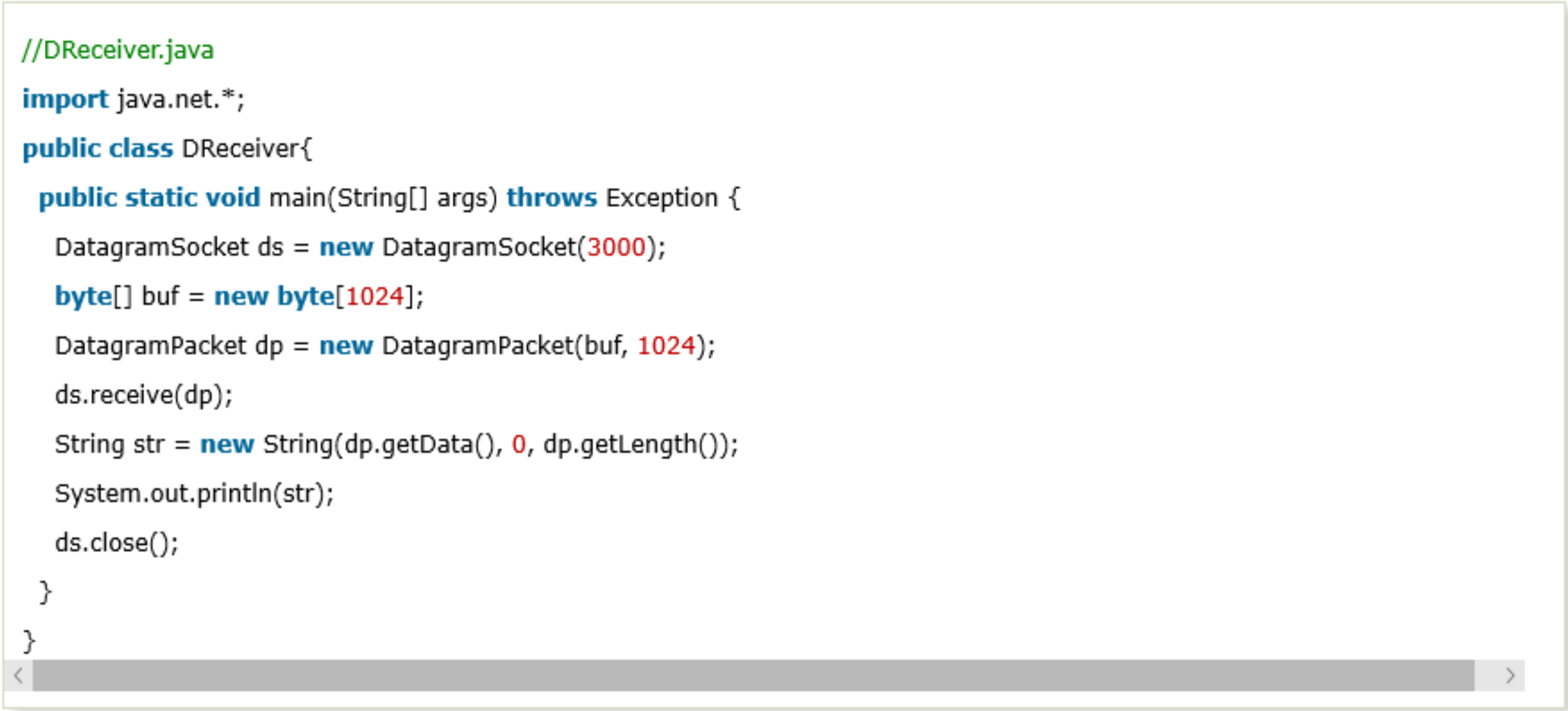
- Commonly used Constructors of DatagramSocket class
  - DatagramPacket(byte[] barr, int length): it creates a datagram packet. This constructor is used to receive the packets.
  - DatagramPacket(byte[] barr, int length, InetAddress address, int port): it creates a datagram packet. This constructor is used to send the packets.

Example of Sending DatagramPacket by DatagramSocket



```
//DSender.java
import java.net.*;
public class DSender{
  public static void main(String[] args) throws Exception {
    DatagramSocket ds = new DatagramSocket();
    String str = "Welcome java";
    InetAddress ip = InetAddress.getByName("127.0.0.1");

    DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
    ds.send(dp);
    ds.close();
  }
}
```

Example of Receiving DatagramPacket by DatagramSocket



```
//DReceiver.java
import java.net.*;
public class DReceiver{
  public static void main(String[] args) throws Exception {
    DatagramSocket ds = new DatagramSocket(3000);
    byte[] buf = new byte[1024];
    DatagramPacket dp = new DatagramPacket(buf, 1024);
    ds.receive(dp);
    String str = new String(dp.getData(), 0, dp.getLength());
    System.out.println(str);
    ds.close();
  }
}
```

Connectionless Server- Client

- For sending a packet via UDP, we should know 4 things, **the message to send, its length, ip address of destination, port at which destination is listening.**
- Once we know all these things, we can create the socket object for carrying the packets and packets which actually possess the data.
- Invoke send()/receive() call for actually sending/receiving packets.
- Extract the data from the received packet.

Client side implementation

```
// Java program to illustrate Client side
// Implementation using DatagramSocket
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class udpBaseClient_2
{
    public static void main(String args[]) throws IOException
    {
        Scanner sc = new Scanner(System.in);

        // Step 1:Create the socket object for
        // carrying the data.
        DatagramSocket ds = new DatagramSocket();

        InetAddress ip = InetAddress.getLocalHost();
        byte buf[] = null;

        // loop while user not enters "bye"
        while (true)
        {
            String inp = sc.nextLine();

            // convert the String input into the byte array.
            buf = inp.getBytes();

            // Step 2 : Create the datagramPacket for sending
            // the data.
            DatagramPacket DpSend =
                  new DatagramPacket(buf, buf.length, ip, 1234);

            // Step 3 : invoke the send call to actually send
            // the data.
            ds.send(DpSend);

            // break the loop if user enters "bye"
            if (inp.equals("bye"))
                break;
        }
    }
}
```

Output:

```
Hello
I am Client.
...
bye
```

Server side implementation

```
// Java program to illustrate Server side
// Implementation using DatagramSocket
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;

public class udpBaseServer_2
{
    public static void main(String[] args) throws IOException
    {
        // Step 1 : Create a socket to listen at port 1234
        DatagramSocket ds = new DatagramSocket(1234);
        byte[] receive = new byte[65535];

        DatagramPacket DpReceive = null;
        while (true)
        {

            // Step 2 : create a DatgramPacket to receive the data.
            DpReceive = new DatagramPacket(receive, receive.length);

            // Step 3 : revieve the data in byte buffer.
            ds.receive(DpReceive);

            System.out.println("Client:-" + data(receive));

            // Exit the server if the client sends "bye"
            if (data(receive).toString().equals("bye"))
            {
                System.out.println("Client sent bye.....EXITING");
                break;
            }

            // Clear the buffer after every message.
            receive = new byte[65535];
        }
    }
```

```
    // A utility method to convert the byte array
    // data into a string representation.
    public static StringBuilder data(byte[] a)
    {
        if (a == null)
            return null;
        StringBuilder ret = new StringBuilder();
        int i = 0;
        while (a[i] != 0)
        {
            ret.append((char) a[i]);
            i++;
        }
        return ret;
    }
}
```

Output:

```
Client:- Hello
Client:- I am client.
...
Client:- bye
Client sent bye.....EXITING
```

Client-Server connectionless

- In order to test the above programs on the system, make sure that you run the server program first and then the client one.
- Make sure you are in the client console and from there keep on typing your messages each followed with a carriage return.
- Finally to terminate the communication, type "bye" (without quotes) and hit enter.
- You should also try and implement a two way chat application wherein the server will be able to respond to messages as and when he likes.