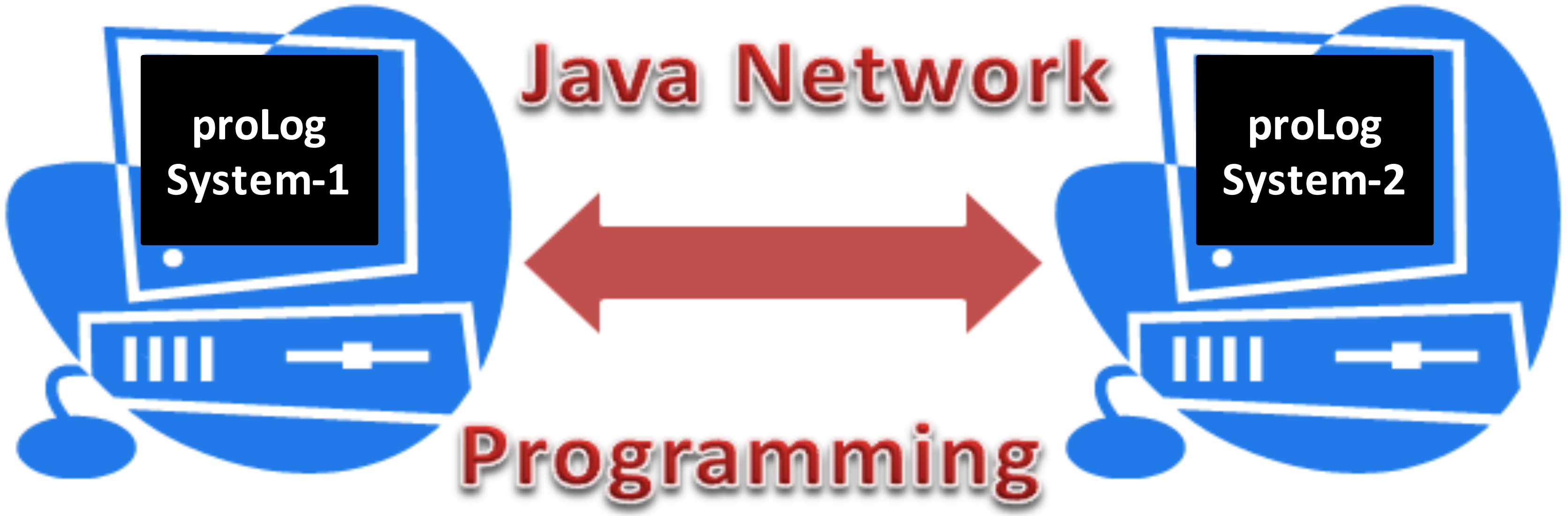
Introduction

- Java supports **Network Programming** to communicate with other machines.
- Let`s start with Network Programming Introduction.



Network Programming

- As we all know that Computer Network means a group of computers connect with each other via some medium and transfer data between them as and when require.

- Java supports Network Programming so we can make such program in which the machines connected in network will send and receive data from other machine in the network by programming.

- The first and simple logic to send or receive any kind of data or message is we must have the address of receiver or sender. So when a computer needs to communicate with another computer, it`s require the other computer’s address.

- Java networking programming supports the concept of socket. A socket identifies an endpoint in a network. The socket communication takes place via a protocol.

Network Programming

- The Internet Protocol is a lower-level, connection less (means there is no continuing connection between the end points) protocol for delivering the data into small packets from one computer (address) to another computer (address) across the network (Internet). It does not guarantee to deliver sent packets to the destination.

- The most widely use a version of IP today is IPv4, uses a 32 bit value to represent an address which are organized into four 8-bits chunks. However new addressing scheme called IPv6, uses a 128 bit value to represent an address which are organized into four 16-bits chunks. The main advantage of IPv6 is that it supports much larger address space than does IPv4. An IP (Internet Protocol) address uniquely identifies the computer on the network.

- IP addresses are written in a notation using numbers separated by dots is called dotted-decimal notation. There are four 8 bits value between 0 and 255 are available in each IP address such as 127.0.0.1 means local-host, 192.168.0.3 etc.

Network Programming

- It`s not an easy to remember because of so many numbers, they are often mapped to meaningful names called domain names such as mail.google.com There is a server on Internet who translate the host names into IP addresses is called DNS (Domain Name Server).

- **NOTE:** Internet is the global network of millions of computer and the any computer may connect the Internet through LAN (Local Area Network), Cable Modem, ISP (Internet Service Provider) using dialup.

- When a user pass the URL like prologacademy.com in the web-browser from any computer, it first ask to DNS to translate this domain name into the numeric IP address and then sends the request to this IP address. This enables users to work with domain names, but the internet operates on IP addresses.

- Here in prologacademy.com the “com” domain is reserved for commercial sites; then “prologacademy” is the company name.

Network Programming

- The Higher-level protocol used in with the IP are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

- The TCP enables two host to make a connection and exchange the stream of data, so it`s called Stream-based communication. TCP guarantees delivery of data and also guarantees that streams of data will be delivered in the same order in which they are sent. The TCP can detect the lost of transmission and so resubmit them and hence the transmissions are lossless and reliable.

- The UDP is a standard, directly to support fast, connectionless host-to-host datagram oriented model that is used over the IP and exchange the packet of data so it`s called packet-based communication. The UDP cannot guarantee lossless transmission.

- JAVA supports both TCP and UDP protocol families.

Java InetAddress Class

Java InetAddress Class is used to encapsulate the two thing.

1. Numeric IP Address
2. The domain name for that address.

- The InetAddress can handle both IPv4 and IPv6 addresses. It has no visible constructors so to create its object, the user have to use one of the available in-built static methods.

The commonly used InetAddress in-built methods are:

(1) getLocalHost():

It returns the InetAddress object that represents the local host contain the name and address both. If this method unable to find out the host name, it throw an UnknownHostException.

Syntax:

Static InetAddress getLocalHost() throws UnknownHostException

(2) getByName():

It returns an InetAddress for a host name passed to it as a parameter argument. If this method unable to find out the host name, it throw an UnknownHostException.

Syntax:

```
Static  InetAddress getByName(String host_name) throwsUnknownHostException
```

(3) getAllByName():

It returns an array of an InetAddress that represent all of the addresses that a particular name resolve to it. If this method can't find out the name to at least one address, it throw an UnknownHostException.

Syntax:

```
Static  InetAddress[] getAllByName(String host_name) throws UnknownHostException
```

Program: Write down a program which demonstrate an InetAddress class.

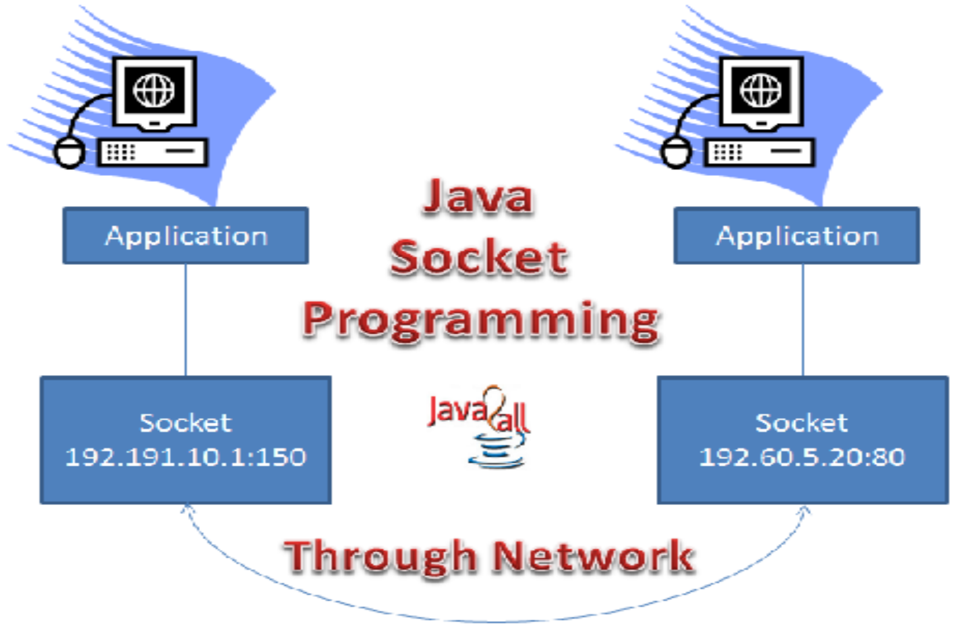
```
import java.net.InetAddress;
import java.net.UnknownHostException;
public class InetAddress_Demo
{
    public static void main(String[] args)
    {
        String name = "";
        try
        {
            S.o.p("HOST NAME - Numeric Address : "+InetAddress.getLocalHost());
            InetAddress ip = InetAddress.getByName(name);
            System.out.println("HOST DEFAULT-NAME / IP : "+ip);
            System.out.println("HOST IP-ADDRESS : "+ip.getHostAddress());
            System.out.println("HOST DEFAULT-NAME : "+ip.getHostName());

        }
        catch (UnknownHostException e)
        {
            System.out.println("Not find the IP-ADDRESS for :"+name);
        }
    }
}
```

Socket Programming in java

socket programming in java is very important topic and concept of network programming.

Java network Programming supports the concept of socket. A socket identifies an endpoint in a network. The socket communication take place via a protocol.

A socket can be used to connect JAVA Input/Output system to other programs that may resides either on any machine on the Internet or on the local machine.



TCP/IP Sockets:

TCP/IP sockets are used to implement point-to-point, reliable, bidirectional, stream-based connections between hosts on the Internet.

There are two types of TCP sockets available in java:

1. TCP/IP Client Socket
2. TCP/IP Server Socket

1. TCP/IP Client Socket:

   The Socket class (available in java.net package) is for the Client Socket. It is designed to connect to server sockets and initiate protocol exchange. There are two constructers used to create client sockets type object.

a. Socket(String host_name,int port) throws UnknownHostException,IOException

It creates a socket that is connected to the given host_name and port number.

b. Socket(InetAddress ip,int port) throws IOException

It creates a socket using a pre-existing InetAddress object and a port number.

2. **TCP/IP Server Socket:**

The ServerSocket class (available in java.net package) is for the Server. It is designed to be a "listener", which waits for clients to connect before doing anything and that listen for either local or remote client programs to connect to them on given port.

When you create ServerSocket it will register itself with the system as having an interest in client connection.

Syntax:

ServerSocket(int port) throws IOException

Program: Write down a program which demonstrate the Socket programming for passing the message from server to client.

Client.java:

Network Programming

Client.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.UnknownHostException;
public class Client
{
	public static void main(String[] args)
	{
		System.out.println("Sending a request.....");
		try {
			Socket s = new Socket("127.0.0.1",1564);
			System.out.println("connected successfully.....");
			BR br = new BR(newInputStreamReader(s.getInputStream()));
			System.out.println("response from server...");
			System.out.println("Client side : "+br.readLine());
			s.close();
		}
		catch (UnknownHostException e)			{
			System.out.println("Not find the IP-ADDRESS for :"+e);
		}
		catch (IOException e)			{
			System.out.println("Not Found data for Socket : "+e);
		}
	}
}
```

Network Programming

Server.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
public class Server
{
        public static void main(String[] args)
        {
        try     {
                                ServerSocket ss = new ServerSocket(1564);
                                System.out.println("waiting for request....");
                                Socket s = ss.accept();
                                System.out.println("Request accepted");
                                PrintStream ps = new PrintStream(s.getOutputStream());
                                BR br = new BR(new InputStreamReader(System.in));
                                System.out.println("Input the data at server : ");
                                ps.print(br.readLine());
                                s.close();
                                ss.close();
                }
                catch (Exception e)     {
                                System.out.println("Not Found data for Socket : "+e);
                }
        }
}
```

For Output follow the below step:

(1) Run server.java

```
Console:
waiting for request....
```

(2) Run Client.java

```
Console:
waiting for request....
Request accepted

Input the data at server:
```

(3) Now enter the message at console

```
Input the data at server:
welcome at server
```

(4) Then press Enter.

(5) Sending a request.....

```
connected successfully.....
response from server...
Client side: welcome at server
```

Program:

Write down a program for addition the two different variable by Socket programming.

Network Programming

Client_Addition.java

```
import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;
public class Client_Addition   {
      public static void main(String[]  args)
      {
            try {
                        Socket s = new Socket("127.0.0.1",1868);
                        PrintStream ps = new PrintStream(s.getOutputStream());
                        Scanner  sc = new Scanner(System.in);
                        System.out.println("Enter  first value:  ");
                        int i1  = sc.nextInt();
                        ps.println(i1);
                        ps.flush();
                        System.out.println("Enter  second  value:  ");
                        int i2  = sc.nextInt();
                        ps.println(i2);
                        ps.flush();
                        s.close();
            }
            catch  (UnknownHostException  e)          {
                                    System.out.println("Not  find the IP-ADDRESS  for :"+e);
            }
            catch  (IOException  e)                         {
                        System.out.println("Not  Found data for Socket : "+e);
            }
      }
```

Server_Addition.java

```
import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;
public class Server_Addition
{
    public static void main(String[] args)
    {
        try
        {
                System.out.println("Server run successfully......”
                ServerSocket sc = new ServerSocket(1868);
                Socket s = sc.accept();
                BR br = new BR(new InputStreamReader(s.getInputStream()));
                int i1 = Integer.parseInt(br.readLine());
                int i2 = Integer.parseInt(br.readLine());
                System.out.println("Addition: "+(i1+i2));
                s.close();
                sc.close();
        }
        catch (IOException e)   {
        System.out.println("Not Found data for Socket : "+e);
        }
    }
}
```

Network Programming

For Output follow the below step:
(1) Run Server_Addition.java

Console:

```
Server run successfully......
```

(2) Run Client.java

Console:

```
Enter first value:
5
Enter second value:
25
```

(3) Now, press Enter

(4) Server run successfully......
Addition: 30

Program:

Write down a program which demonstrate the Socket programming for passing the message from client to server and also apply EXIT properties.

Client1.java

```
import java.io.*;
import java.net.*;
public class Client1  {
    public static void main(String[] args)  {
        System.out.println("Sending a request.....");
        try {
            Socket s = new Socket("127.0.0.1",1235);
            System.out.println("connected successfully.....");
            BR br = new BR(new InputStreamReader(System.in));
            PrintStream ps = new PrintStream(s.getOutputStream());
            BufferedReader brs = new BufferedReader(new
            InputStreamReader(s.getInputStream()));
            while(true)    {
                    System.out.println("input the data....");
                    String st = br.readLine();
                    ps.println(st);
                    if(st.equals("exit"))   {
                        System.exit(1);

                    }
                System.out.println("data returned");
                System.out.println(st);
            }
        }
        catch (UnknownHostException e)      {
            System.out.println("Not find the IP-ADDRESS for :"+e);
        }
        catch (IOException e)      {
            System.out.println("Not Found data for Socket : "+e);
        }
    }
}
```

Network Programming

Server1.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
public class Server1 {
	public static void main(String[] args)
	{
		try
		{
			ServerSocket ss = new ServerSocket(1235);
			System.out.println("waiting for request....");
			Socket s = ss.accept();
			System.out.println("Request accepted");
			BR br = new BR(new InputStreamReader(s.getInputStream()));
			while(true)
			{
				String st = br.readLine();
				if(st.equals("exit")==true)
				{
					System.out.println("connection lost.....");
					System.exit(1);
				}
				System.out.println("Message from client: "+st);
			}
		}
		catch (IOException e)
		{
			System.out.println("Not Found data for Socket : "+e);
		}
	}
}
```

For Output follow the below step:

(1) Put the both file in the bin folder at jdk.

For example: C:\Program Files (x86)\Java\jdk1.6.0\bin.

(2) Open Command Prompt & reach up to bin path
(3) Compile the Server.java & Client.java

```
...\bin>javac Server.java

...\bin>javac Client.java
```

(4) Run the Server.java

```
...\bin>java Server
```

(5) Open new command prompt:
(6) Now revise step-2.
(7) Run the Client.java.

```
...\bin>java Client
```

Check the Message at Server Side Command Prompt.

(8) Write down the message on Client Side Command Prompt Like:

```
Input the data...
Tausif
```

(9) Now Press Enter & Check the Output at Both Windows.
(10) If want to Exit then type exit on Client side Window.
Like: Input the data...
exit