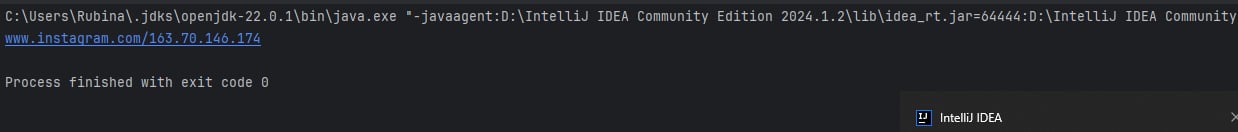
* 1. **Write the program that prints the address of** [**www.Instagram.com**](http://www.Instagram.com)**.**

**Program:**

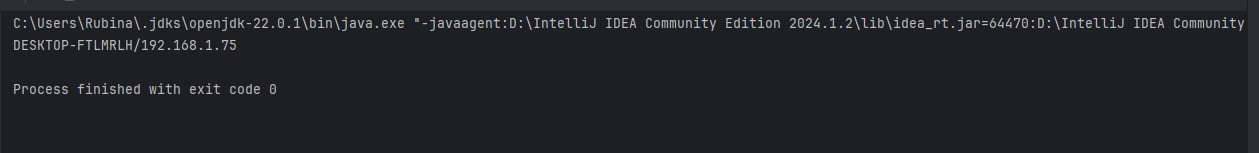
import java.net.InetAddress;  
import java.net.UnknownHostException;  
import java.util.Scanner;  
  
public class DomainClass {  
 public static void main(String[] args){   
 try{  
 InetAddress address=InetAddress.getByName(“www.instagram.com”);  
 System.out.println(address);  
 }  
 catch (UnknownHostException e){  
 System.out.println("Invalid "+url);  
 }  
 }  
}

**Output:**

* 1. **Write a program to find the address of the local machine.**

**Program:**

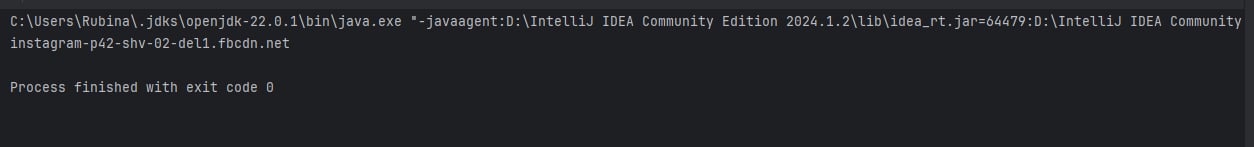
import java.net.InetAddress;  
import java.net.UnknownHostException;  
  
public class LocalAddressProgram {  
 public static void main(String[] args){  
 try{  
 InetAddress localIp=InetAddress.getLocalHost();  
 System.out.println(localIp);  
 }  
 catch (UnknownHostException e){  
 System.out.println("Man!! I got no address.");  
 }  
 }  
}

**Output**:

* 1. **Write a program to find the hostname with given address.**

**Program:**

import java.net.InetAddress;  
import java.net.UnknownHostException;  
  
public class IpClass {  
 public static void main(String[] args){  
 try{  
 InetAddress ipAddress=InetAddress.getByName("157.240.239.174");  
 System.out.println(ipAddress.getCanonicalHostName());  
 }  
 catch (UnknownHostException e){  
 System.out.println("Invalid IP Address!");  
 }  
 }  
}

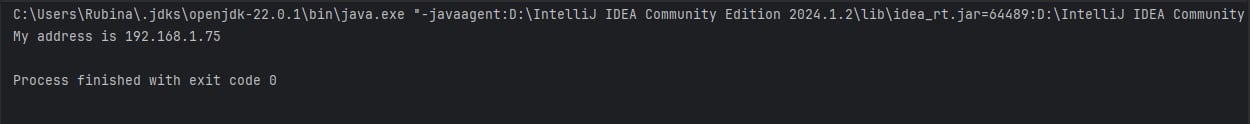
**Output:**

* 1. **Write a program to find the IP address of the local machine.**

**Program:**

import java.net.\*;  
  
public class MyAddressProgram {  
 public static void main(String[] args){  
 try {  
 InetAddress me = InetAddress.getLocalHost();  
 String dottedQuad = me.getHostAddress();  
 System.out.println("My address: " + dottedQuad);  
 } catch (UnknownHostException ex) {  
 System.out.println("Myan! I got no address");  
 }  
 }  
}

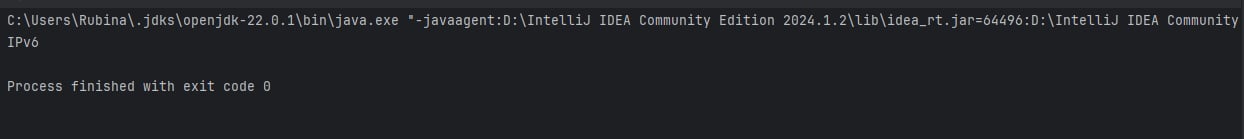
**Output:**

****

* 1. **Write a program to determine whether the address is IPv4 or IPv6.**

**Program:**

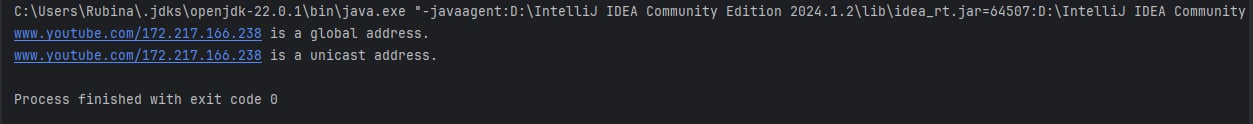
import java.net.\*;  
public class IPVersionProgram {  
 public static void main(String[] args){  
 try {  
 InetAddress ia = InetAddress.getByName("FF00::");  
 byte[] address = ia.getAddress();  
 if(address.length==4)  
 System.out.println(ia + " is IPv4.");  
 else if(address.length==16)  
 System.out.println(ia + " is IPv6.");  
 } catch (UnknownHostException ex) {  
 System.out.println("Invalid IP!");  
 }  
 }  
}

**Output:**

* 1. **Write a program to test the characteristics of an IP address**

**Program:**

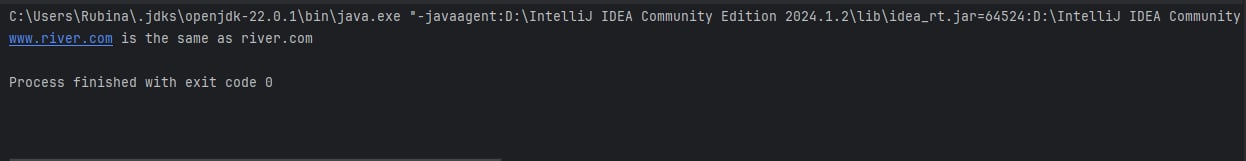
import java.net.\*;  
public class IPCharacteristics {  
 public static void main(String[] args) {  
 try {  
 InetAddress address = InetAddress.getByName("www.Youtube.com");  
 if (address.isAnyLocalAddress()) {  
 System.out.println(address + " is a wildcard address.");  
 }  
 if (address.isLoopbackAddress()) {  
 System.out.println(address + " is a loopback address.");  
 }  
 if (address.isLinkLocalAddress()) {  
 System.out.println(address + " is a link-local address.");  
 } else if (address.isSiteLocalAddress()) {  
 System.out.println(address + " is a site-local address.");  
 } else {  
 System.out.println(address + " is a global address.");  
 }  
 if (address.isMulticastAddress()) {  
 if (address.isMCGlobal()) {  
 System.out.println(address + " is a global multicast address.");  
 } else if (address.isMCOrgLocal()) {  
 System.out.println(address + " is an organization wide multicast address.");  
 } else if (address.isMCSiteLocal()) {  
 System.out.println(address + " is a site wide multicast address.");  
 } else if (address.isMCLinkLocal()) {  
 System.out.println(address + " is a subnet wide multicast address.");  
 } else if (address.isMCNodeLocal()) {  
 System.out.println(address + " is an interface-local multicast address.");  
 } else {  
 System.out.println(address + " is an unknown multicast address type.");  
 }  
 } else {  
 System.out.println(address + " is a unicast address.");  
 }  
 } catch (UnknownHostException ex) {  
 System.err.println("Could not resolve the address.");  
 }  
 }  
}

**Output:**

* 1. **Write a Program to determine where two different domain name have same IP address.**

**Program:**

import java.net.\*;  
  
public class SameIPProgram {  
 public static void main(String[] args) {  
 try {  
 InetAddress site1 = InetAddress.getByName("www.river.com");  
 InetAddress site2 = InetAddress.getByName("river.com");  
 if (site1.equals(site2)) {  
 System.out.println(“www.example.com is the same as example.com”);  
 } else {  
 System.out.println("Different Site");  
 }  
 } catch (UnknownHostException ex) {  
 System.out.println("Host lookup failed.");  
 }  
 }  
}

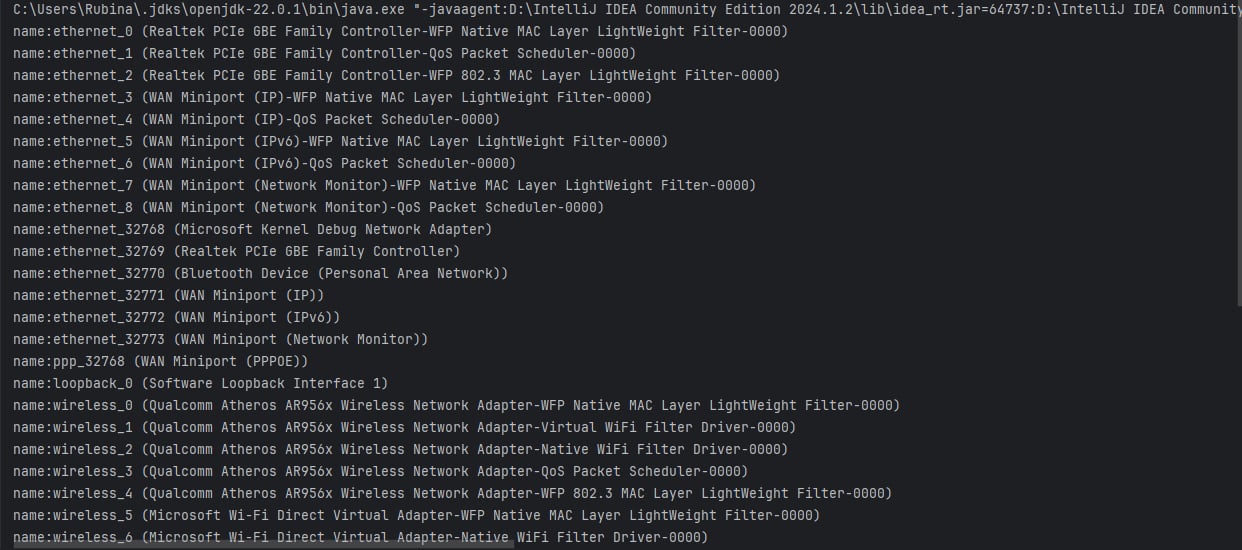
**Output:**

* 1. **Write a program that lists all the network interfaces.**

**Program:**

import java.net.\*;  
import java.util.\*;  
public class InterfaceListener {  
 public static void main(String[] args) throws SocketException {  
 Enumeration<NetworkInterface> interfaces = NetworkInterface.getNetworkInterfaces();  
 while (interfaces.hasMoreElements()) {  
 NetworkInterface ni = interfaces.nextElement();  
 System.out.println(ni);  
 }  
 }

}

**Output:**

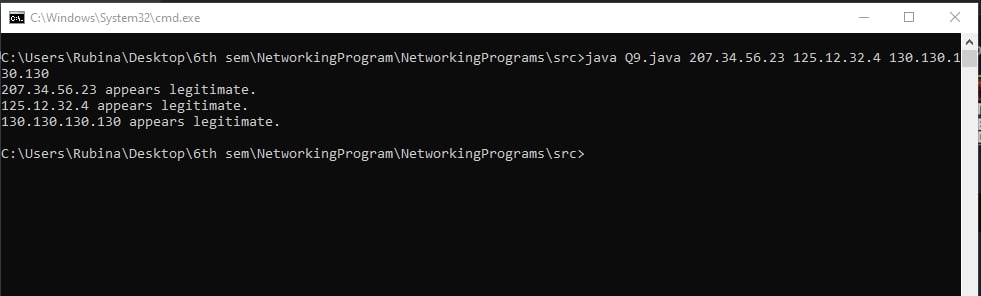
* 1. **Write a program to demonstrate SpamCheck.**

**Program:**

import java.net.\*;  
  
public class SpamCheck {  
  
 public static final String BLACKHOLE = "sbl.spamhaus.org";  
  
 public static void main(String[] args) throws UnknownHostException {  
 for (String arg: args) {  
 if (isSpammer(arg)) {  
 System.out.println(arg + " is a known spammer.");  
 } else {  
 System.out.println(arg + " appears legitimate.");  
 }  
 }  
 }  
  
 private static boolean isSpammer(String arg) {  
 try {  
 InetAddress address = InetAddress.getByName(arg);  
 byte[] quad = address.getAddress();  
 String query = BLACKHOLE;  
 for (byte octet : quad) {  
 int unsignedByte = octet < 0 ? octet + 256 : octet;  
 query = unsignedByte + "." + query;  
 }  
 InetAddress.getByName(query);  
 return true;  
 } catch (UnknownHostException e) {  
 return false;  
 }

}  
}

**Output:**

****