MODULE V

Basics of Java Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Types of the applet

- Applets based on the **AWT(Abstract Window Toolkit)** package by extending its **Applet** class.
- Applets based on the **Swing** package by extending its **JApplet** class

Some methods of Applet class

Methods	Description
void init()	The first method to be called when an applet begins its execution.
void start()	Called automatically after **init()** method to start the execution of an applet.
void stop()	Called when an applet has to pause/stop its execution.
void destoy()	Called when an appLet us finally terminated.
String getParameter(String ParameterName)	Returns the value of a parameter defined in an applet
Image getImage(URL *url*)	Returns an Image object that contains an image specified at location, *url*
void play(URL *url*	Plays an audio clip found at the specified at location, *url*
showStatus(String str)	Shows a message in the status window of the browser or appletviewer.

Advantage of Applet

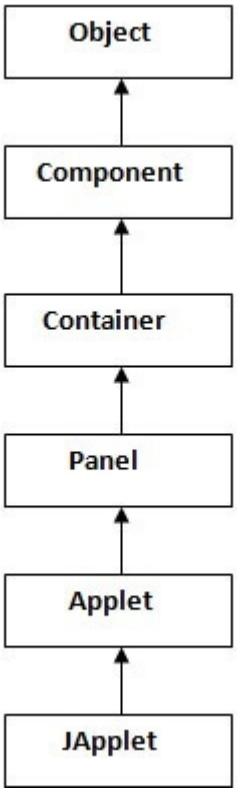
There are many advantages of applet. They are as follows:

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

- Plugin is required at client browser to execute applet.

Hierarchy of Applet



As displayed in the above diagram, Applet class extends Panel. Panel class extends Container which is the subclass of Component.

Lifecycle of Java Applet

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.

5. Applet is destroyed.

Lifecycle methods for Applet:

The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods for an applet.

java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides 4 life cycle methods of applet.

It is important to understand the order in which the various methods shown in the above image are called. When an applet begins, the following methods are called, in this sequence:

1. init( )
2. start( )
3. paint( )

When an applet is terminated, the following sequence of method calls takes place:

1. stop( )
2. destroy( )

Let's look more closely at these methods.

1. **init( ) :** The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called **only once** during the run time of your applet.
2. **start( ) :** The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Note that **init( )** is called once i.e. when the first time an applet is loaded whereas **start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.
3. **paint( ) :** The **paint( )** method is called each time an AWT-based applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored. **paint( )** is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )** is called. The **paint( )** method has one parameter of type Graphics. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

4. **stop( ) :** The **stop( )** method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example. When **stop( )** is called, the applet is probably running. You should use **stop( )** to suspend threads that don't need to run when the applet is not visible. You can restart them when **start( )** is called if the user returns to the page.
5. **destroy( ) :** The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory. At this point, you should free up any resources the applet may be using. The **stop( )** method is always called before **destroy( )**.

java.awt.Component class

The Component class provides 1 life cycle method of applet.

1. **public void paint(Graphics g):** is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

Responsible to manage the life cycle of an applet

Java Plug-in software.

To run an Applet

There are two ways to run an applet

1. By html file.

2. By appletViewer tool (for testing purpose).

Simple example of Applet by html file:

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet code in html file. Now click the html file.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome",150,150);
}
  }
```

myapplet.html

```
<html>
<body>
<applet code="First.class" width="300" height="300">
</applet>
</body>
</html>
```

Simple example of Applet by appletviewer tool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html file is not required but it is for testing purpose only.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{
  public void paint(Graphics g){
g.drawString("welcome to applet",150,150);
}
  }
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

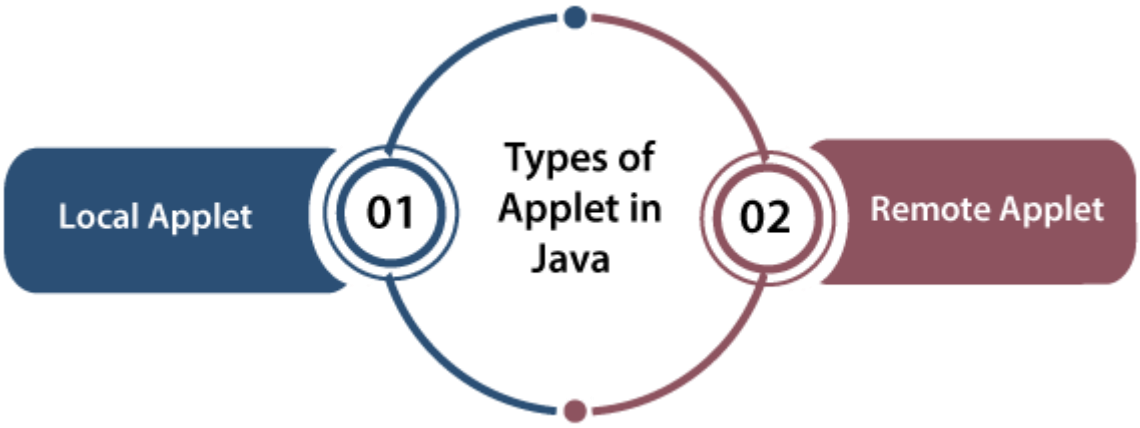
To execute the applet by appletviewer tool, write in command prompt:

```
c:\>javac First.java
c:\>appletviewer First.java
```

Types of Applets in Java

A special type of Java program that runs in a Web browser is referred to as **Applet**. It has less response time because it works on the client-side. It is much secured executed by the browser under any of the platforms such as Windows, Linux and Mac OS etc. There are two types of applets that a web page can contain.

1. **Local Applet**
2. **Remote Applet**



1.Local Applet

Local Applet is written on our own, and then we will embed it into web pages. Local Applet is developed locally and stored in the local system. A web page doesn't need the get the information from the internet when it finds the local Applet in the system. It is specified or defined by the file name or pathname. There are two attributes used in defining an applet, i.e., the **codebase** that specifies the path name and **code** that defined the name of the file that contains Applet's code.

Specifying Local applet

```
<applet
   codebase = "tictactoe"
   code = "FaceApplet.class"
   width = 120
   height = 120>
</applet>
```

Let's take an example of Local applet to understand how we can create it and embedded it into web page.

1. First, we will create a Local Applet for embedding in a web page.
2. After that, we will add that Local Applet to the web page.

FaceApplet.java

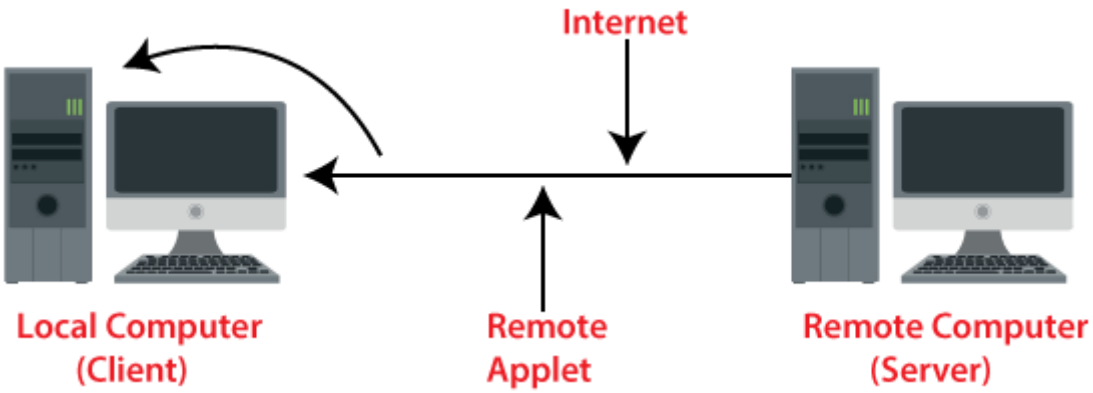
```
//Import packages and classes
import java.applet.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
//Creating FaceApplet class that extends Applet
public class FaceApplet extends Applet
{
   //paint() method starts
   public void paint(Graphics g){
      //Creating graphical object
      g.setColor(Color.red);
      g.drawString("Welcome", 50, 50);
      g.drawLine(20, 30, 20, 300);
      g.drawRect(70, 100, 30, 30);
      g.fillRect(170, 100, 30, 30);
      g.drawOval(70, 200, 30, 30);
      g.setColor(Color.pink);
      g.fillOval(170, 200, 30, 30);
      g.drawArc(90, 150, 30, 30, 30, 270);
      g.fillArc(270, 150, 30, 30, 0, 180);
   }  }
```

Execute the above code by using the following commands:

2.Remote Applet

A remote applet is designed and developed by another developer. It is located or available on a remote computer that is connected to the internet. In order to run the applet stored in the remote computer, our system is connected to the internet then we can download run it. In order

to locate and load a remote applet, we must know the applet's address on the web that is referred to as Uniform Recourse Locator(URL).



Specifying Remote applet

```
<applet
   codebase = "http://www.myconnect.com/applets/"
   code = "FaceApplet.class"
   width = 120
   height =120>
</applet>
```

Difference Between Local Applet and Remote Applet

The following table describes the key differences between Local applet and Remote applet.

Local Applet	Remote Applet
There is no need to define the Applet's URL in Local Applet.	We need to define the Applet's URL in Remote Applet.
Local Applet is available on our computer.	Remote Applet is not available on our computer.
In order to use it or access it, we don't need Internet Connection.	In order to use it or access it on our computer, we need an Internet Connection.
It is written on our own and then embedded into the web pages.	It was written by another developer.

We don't need to download it.	It is available on a remote computer, so we need to download it to our system.

Difference between a Java Application and a Java Applet

Java Application is just like a Java program that runs on an underlying operating system with the support of a virtual machine. It is also known as an application program. The graphical user interface is not necessary to execute the java applications, it can be run with or without it.

Java Applet is an applet is a Java program that can be embedded into a web page. It runs inside the web browser and works on the client-side. An applet is embedded in an HTML page using the **APPLET** or **OBJECT** tag and hosted on a web server. Applets are used to make the website more dynamic and entertaining.

Parameters	Java Application	Java Applet
Definition	Applications are just like a Java program that can be executed independently without using the web browser.	Applets are small Java programs that are designed to be included with the HTML web document. They require a Java-enabled web browser for execution.
main () method	The application program requires a main() method for its execution.	The applet does not require the main() method for its execution instead init() method is required.
Compilation	The “javac” command is used to compile application programs, which are then executed using the “java” command.	Applet programs are compiled with the “javac” command and run using either the “appletviewer” command or the web browser.
File access	Java application programs have full access to the local file system and network.	Applets don’t have local disk and network access.

Access level	Applications can access all kinds of resources available on the system.	Applets can only access browser-specific services. They don't have access to the local system.
Installation	First and foremost, the installation of a Java application on the local computer is required.	The Java applet does not need to be installed beforehand.
Execution	Applications can execute the programs from the local system.	Applets cannot execute programs from the local machine.
Program	An application program is needed to perform some tasks directly for the user.	An applet program is needed to perform small tasks or part of them.
Run	It cannot run on its own; it needs JRE to execute.	It cannot start on its own, but it can be executed using a Java-enabled web browser.
Connection with servers	Connectivity with other servers is possible.	It is unable to connect to other servers.
Read and Write Operation	It supports the reading and writing of files on the local computer.	It does not support the reading and writing of files on the local computer.
Security	Application can access the system's data and resources without any security limitations.	Executed in a more restricted environment with tighter security. They can only use services that are exclusive to their browser.
Restrictions	Java applications are self-contained and require no additional security because they are trusted.	Applet programs cannot run on their own, necessitating the maximum level of security.

Java Networking

Java Networking is a concept of connecting two or more computing devices together so that we can share resources.

Java socket programming provides facility to share data between different computing devices.

Advantage of Java Networking

1. Sharing resources
2. Centralize software management

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.

5. **Socket –** A socket is one endpoint of a two-way communication connection between the two applications running on the network. The socket mechanism presents a method of inter-process communication (IPC) by setting named contact points between which the communication occurs. A socket is tied to a port 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)

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 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 comprises a CookieStore and a CookiePolicy.

4. **DatagramPacket –** The DatagramPacket class is used to provide a facility for the connectionless transfer of messages from one system to another. This class 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.

10. **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, or it can indicate a more difficult object, such as a query to a database or a search engine.

11. **URLConnection –** The URLConnection class in Java is an abstract class describing 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 implementations that implement various policies.

6. **ProtocolFamily –** This interface represents a family of communication protocols. The ProtocolFamily interface contains a method known as name(), which returns the name of the protocol family.

java.net package

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

1. **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.
2. **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.

TCP/IP Client Sockets

TCP/IP sockets are used to implement reliable two-way, persistent, point-to-point streaming connections between hosts on the Internet. The Java I/O system can use sockets to connect to other programs on the local system or on other systems on the Internet. It is important to note that the applet establishes a reverse socket connection to the host on which the applet is loaded. This restriction exists because it is dangerous for applets loaded through a firewall to access arbitrary systems. There are two types of TCP sockets in Java.

One for the **server** and one for the **client**. The ServerSocket class is designed as a "listener", waiting for a client to connect before doing anything. So ServerSocket is for servers. The Socket class is for clients. It is designed to connect to a server socket and initiate a protocol exchange. This is because client sockets are most commonly used in Java applications. Creating a Socket object implicitly establishes a connection between the client and server. There is no method or constructor that explicitly exposes details about setting up this connection.

Here are the two constructors used to create a client socket:

1. **Socket(String hostName, int port) throws UnknownHostException, IOException:** Creates a socket connected to the specified host and port.
2. **Socket(InetAddress ipAddress, int port) throws IOException:** Creates a socket using a pre-existing InetAddress object and a port.

Socket defines multiple instance methods. For example, a Socket can always check for associated address and port information using the following methods:

1. **InetAddress getInetAddress( ):** It returns the InetAddress associated with the Socket object. It returns null if the socket is not connected.
2. **int getPort( ):** It returns the remote port to which the invoking Socket object is connected. It returns 0 if the socket is not connected.

3. **int getLocalPort( ):** Returns the local port to which the invoking Socket object is bound. It returns -1 if the socket is not bound.
4. **InputStream getInputStream( ) throws IOException:** Returns the InputStream associated with the invoking socket.
5. **OutputStream getOutputStream( ) throws IOException:** Returns the OutputStream associated with the invoking socket.
6. **connect( ):** Allows you to specify a new connection
7. **isConnected( ):** Returns true if the socket is connected to a server
8. **isBound( ):** Returns true if the socket is bound to an address
9. **isClosed( ):** Returns true if the socket is closed.

The following program provides a simple socket example. Opens a connection to a "whois" port (port 43) of the InterNIC server, sends command-line argument to the socket, and prints the returned data. The InterNIC will try find the argument by the registered Internet domain name, and then send back the IP address and contact information for that site.

Example of Stream Socket

WhoisClient.java

```
1.  import java.net.*;
2.  import java.io.*;
3.  public class WhoisClient {
4.      public static void main(String[] args) {
5.          // no arguments passed, simply return
6.          if (args.length < 1)
7.          return;
8.          // initializing domainName with the name passed in the argument
9.          String domainName = args[0];
10.         // specifying the host name
11.         String hostname = "whois.internic.net";
12.         int port = 43;
13.         try (Socket socket = new Socket(hostname, port)) {
14.             // getOutputStream( ) returns the OutputStream
15.             // associated with the invoking socket
16.             OutputStream output = socket.getOutputStream();
17.             PrintWriter writer = new PrintWriter(output, true);
18.             // print the domain name
19.             writer.println(domainName);
20.             // getInputStream( ) returns the InputStream
21.             // associated with the invoking socket
22.             InputStream input = socket.getInputStream();
23.             BufferedReader reader = new BufferedReader(new InputStreamReader(inpu
    t));
24.             String line;
```

```
25.             while ((line = reader.readLine()) != null) {
26.                 System.out.println(line);
27.             }
28.         }
29.         catch (UnknownHostException ex) {
30.             System.out.println("Server not found: " + ex.getMessage());
31.          }
32.         catch (IOException ex) {
33.             System.out.println("I/O error: " + ex.getMessage());
34.         }
35.     }
36. }
```

Output:

```
D:\JavaPrograms>java WhoisClient wikipedia.com
   Domain Name: WIKIPEDIA.COM
   Registry Domain ID: 51687032_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.markmonitor.com
   Registrar URL: http://www.markmonitor.com
   Updated Date: 2021-12-09T09:20:37Z
   Creation Date: 2001-01-13T00:12:14Z
   Registry Expiry Date: 2023-01-10T05:28:20Z
   Registrar: MarkMonitor Inc.
   Registrar IANA ID: 292
```

Here is how the program works. First, the socket is constructed to compile the host name "internic.net" and the port number 43, which is an online site that handles the port 43 whois requests. In addition, both input and output streams are opened on the socket. Then, the string is constructed that contains the name of the web site we want to get information. The response is read by inputting from the socket, and the results are displayed.

TCP/IP Client Sockets

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet. A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

y are examined here.

The creation of a Socket object implicitly establishes a connection between the client

and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets:

Socket(String *hostName*, int *port*) throws UnknownHostException, IOException	Creates a socket connected to the named host and port.
Socket(InetAddress *ipAddress*, int *port*) throws IOException	Creates a socket using a preexisting **InetAddress** object and a port.

Socket defines several instance methods. For example, a Socket can be examined at any time for the address and port information associated with it, by use of the following methods:

InetAddress getInetAddress( )	Returns the **InetAddress** associated with the **Socket** object. It returns null if the socket is not connected.
int getPort( )	Returns the remote port to which the invoking **Socket** object is connected. It returns 0 if the socket is not connected.
int getLocalPort( )	Returns the local port to which the invoking **Socket** object is bound. It returns –1 if the socket is not bound.

You can gain access to the input and output streams associated with a Socket by use of the getInputStream( ) and getOuptutStream( ) methods, as shown here. Each can throw an IOException if the socket has been invalidated by a loss of connection. These streams are used exactly like the I/O streams to send and receive data.

InputStream getInputStream( ) throws IOException	Returns the **InputStream** associated with the invoking socket.
OutputStream getOutputStream( ) throws IOException	Returns the **OutputStream** associated with the invoking socket.

Several other methods are available, including connect( ), which allows you to specify a new connection; isConnected( ), which returns true if the socket is connected to a server; isBound( ),

which returns true if the socket is bound to an address; and isClosed( ), which returns true if the socket is closed.

The following program provides a simple Socket example. It opens a connection to a "whois" port (port 43) on the InterNIC server, sends the command-line argument down the socket, and then prints the data that is returned. InterNIC will try to look up the argument as a registered Internet domain name, and then send back the IP address and contact information for that site.

```
// Demonstrate Sockets.
import java.net.*;
import java.io.*;
class Whois {
public static void main(String args[]) throws Exception {
int c;
// Create a socket connected to internic.net, port 43.
Socket s = new Socket("internic.net", 43);
// Obtain input and output streams.
InputStream in = s.getInputStream();
OutputStream out = s.getOutputStream();
// Construct a request string.
```



```
String str = (args.length == 0 ? "osborne.com" : args[0]) + "\n";
// Convert to bytes.
byte buf[] = str.getBytes();
// Send request.
out.write(buf);
// Read and display response.
while ((c = in.read()) != -1) {
System.out.print((char) c);
}
s.close();
}
}
```

If, for example, you obtained information about osborne.com, you'd get something similar to the following:

```
Whois Server Version 1.3
```

Domain names in the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.
Domain Name: OSBORNE.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: NS1.EPPG.COM
Name Server: NS2.EPPG.COM
.
.
.

Here is how the program works. First, a Socket is constructed that specifies the host name "internic.net" and the port number 43. Internic.net is the InterNIC web site that handles whois requests. Port 43 is the whois port. Next, both input and output streams are opened on the socket. Then, a string is constructed that contains the name of the web site you want to obtain information about. In this case, if no web site is specified on the command line, then "osborne.com" is used. The string is converted into a byte array and then sent out of the socket. The response is read by inputting from the socket, and the results are displayed.