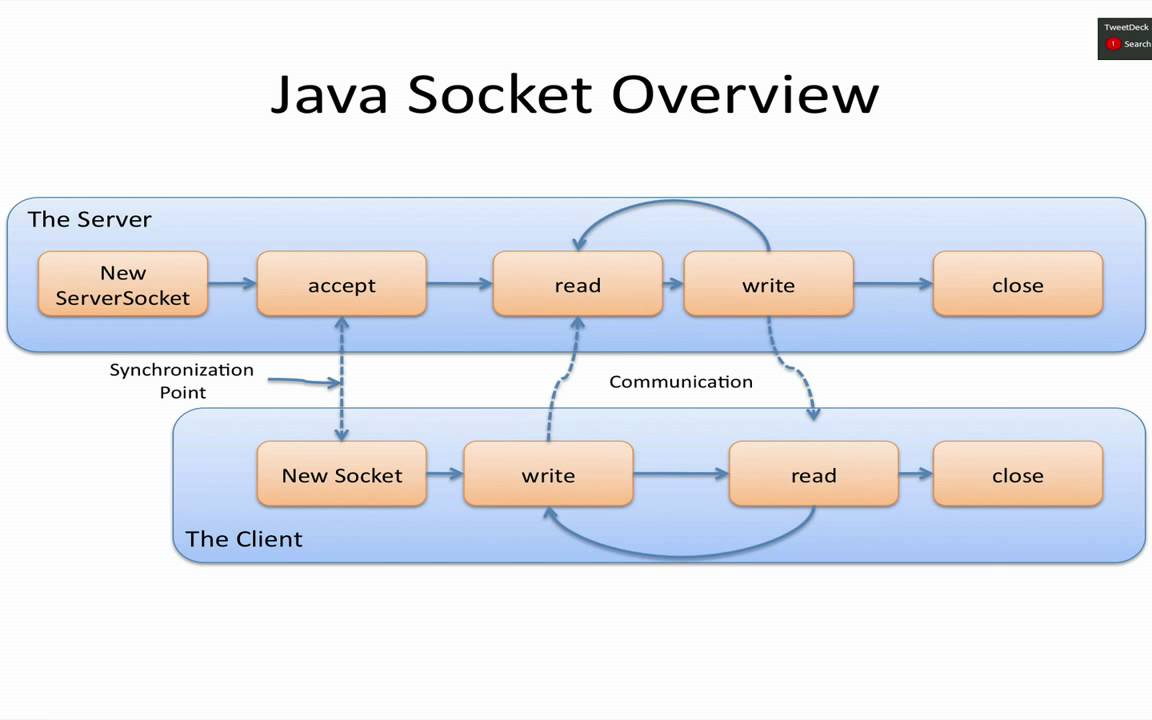
JAVA NETWORK PROGRAMMING

CSE215 | DAFFODIL INTERNATIONAL UNIVERSITY

OBJECT ORIENTED PROGRAMMING LAB

SESSION - 11

11

**JAVA NETWORK PROGRAMMING:**   
The term *network programming* refers to writing programs that execute across multiple devices (computers), in which the devices are all connected to each other using a network.  
  
**java.net package:**  
  
The java.net package provides support for the two common network protocols –

* **TCP** − TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
* **UDP** − UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

**Socket Programming** − Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less. Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

## Socket Class Methods:

The **java.net.Socket** class represents the socket that both the client and the server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept() method.

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **public Socket(InetAddress host, int port) throws IOException  throws UnknownHostException, IOException.**This method attempts to connect to the specified server at the specified port. |
| 2 | **public void connect(SocketAddress host, int timeout) throws IOException** This method connects the socket to the specified host. This method is needed only when you instantiate the Socket using the no-argument constructor. |
| 3 | **public InetAddress getInetAddress()** This method returns the address of the other computer that this socket is connected to. |
| 4 | **public int getPort()** Returns the port the socket is bound to on the remote machine. |
| 5 | **public InputStream getInputStream() throws IOException** Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket. |
| 6 | **public OutputStream getOutputStream() throws IOException** Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket. |

## ServerSocket Class Methods:

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests.

The ServerSocket class has four constructors −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **public ServerSocket(int port) throws IOException** Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application. |
| 2 | **public int getLocalPort()** Returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you. |
| 3 | **public Socket accept() throws IOException** Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely. |
| 4 | **public void setSoTimeout(int timeout)** Sets the time-out value for how long the server socket waits for a client during the accept(). |
| 5 | **public void bind(SocketAddress host, int backlog)** Binds the socket to the specified server and port in the SocketAddress object. Use this method if you have instantiated the ServerSocket using the no-argument constructor. |

## Example 11.1: CLIENT:

1. **import** java.io.\*;
2. **import** java.net.\*;
3. **import** java.util.Date;
4. **import** javax.swing.JOptionPane;
6. **public** **class** DateClient {
8. **public** **static** **void** main(String[] args) **throws** Exception{
9. String svradr=JOptionPane.showInputDialog("Enter IP address of machine that is**\n**"+"running the date service on port 9090:");
10. Socket s= **new** Socket(svradr, 9090);
11. BufferedReader br=**new** BufferedReader(
12. **new** InputStreamReader(
13. s.getInputStream()));
14. String answer= br.readLine();
15. JOptionPane.showMessageDialog(**null**, answer);
16. System.exit(0);
17. }
19. }

**Example 11.2: SERVER:**

1. **import** java.io.\*;
2. **import** java.net.\*;
3. **import** java.util.Date;
5. **public** **class** DateServer {
7. **public** **static** **void** main(String[] args)
8. **throws** Exception{
9. ServerSocket sc=**new** ServerSocket(9090);
10. **try**{
11. **while**(**true**){
12. Socket s=sc.accept();
13. System.out.println("Request from: "+ s.getInetAddress().getHostAddress());
14. PrintWriter out= **new** PrintWriter(s.getOutputStream(), **true**);
15. out.println("Server says: "+**new** Date().toString());
17. s.close();
18. }
19. }
20. **finally**{
21. sc.close();
22. }
23. }
25. }