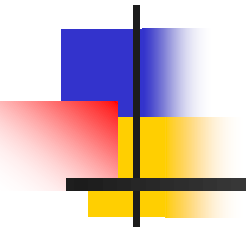


Network Programming

inter-process communication

Sockets

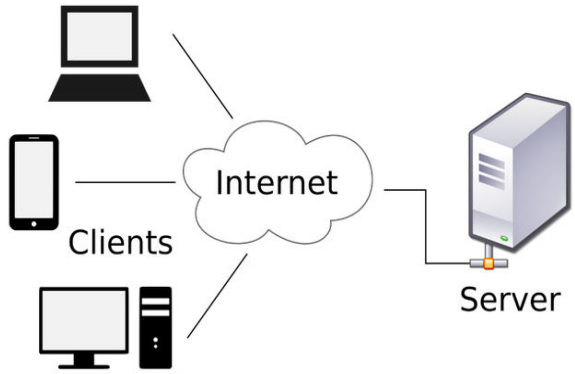
Socket Definition and Types

- **Sockets** are network communication link end-points between two applications (i.e., server and client).

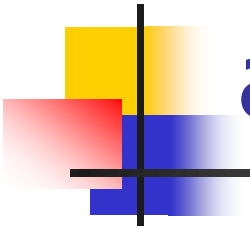
- There are two type of sockets that can be used for application development:

  - **Transport layer sockets:**
    make use of transport-layer protocols.

  - **Application layer sockets:**
    make use of application-layer protocols.

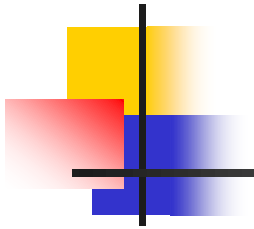


two type of sockets are used for application development



- **Transport layer sockets:** such as:
  - The Transmission Control Protocol (**TCP**). TCP is a connection-based reliable, orderly transmission of data packets, similar to the telephone service.
  - The User Datagram Protocol (**UDP**). UDP is a connectionless non-reliable transmission of datagrams protocol, similar to the postal service.
- **Application layer sockets:** such as:
  - Hypertext Transfer Protocol (**HTTP**). HTTP is a TCP-based web page delivery service
  - Simple Mail Transfer Protocol (**SMTP**). SMTP is a TCP-based e-mail delivery service.

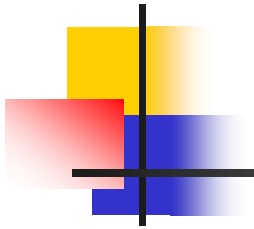
Java Sockets Programming



The *java.net* package provides support for the two common network protocols:

- **TCP:** stands for Transmission Control Protocol.
- **UDP:** stands for User Datagram Protocol.

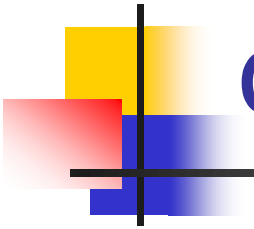
Java Sockets Programming



The *java.net* Java provides three different types of sockets

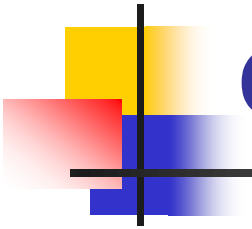
- **Socket** class: it is a **Connection-oriented (TCP)**.
- **Datagramsocket** class: it is a **Connectionless (UDP)**.
- **Multicastsocket** class: it is a subclass of the DatagramSocket class and allows data to be sent to multiple recipients.

Java TCP Client communication steps



- Step 1— Create a TCP client socket and establish a connection to the server.
- Step 2— Set input and output streams.
- Step 3— Send and receive data.
- Step 4— Close the connection.

Java TCP server communication steps



- Step 1—Create a TCP server socket object.
- Step 2—Set the server to wait (block) for clients to connect.
- Step 3—Set input and output streams.
- Step 4—Send and receive data.
- Step 5—Close the connection.

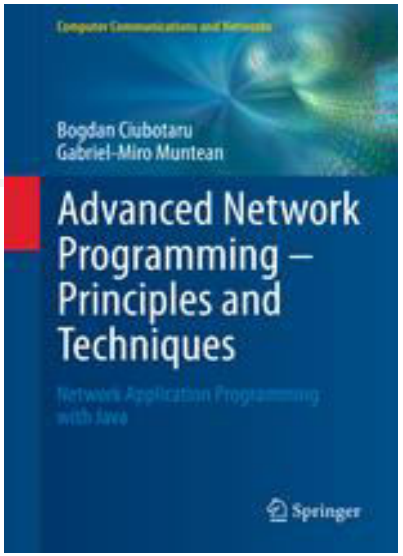
```
import java.net.*;
import java.io.*;                          // example TCP1 Server Program
import java.util.*;
public class NumberServer {
    public static void main (String args[]) throws IOException {
        ServerSocket server = new ServerSocket(4446);
        while (true) {
            System.out.println("Waiting for client...");
             Socket client = server.accept();
            System.out.println("Client connected.");
            DataOutputStream output=null;
            try {
                output = new DataOutputStream(
                        client.getOutputStream() );
            }
            catch (IOException e) {  System.out.println(e);  }
            output.writeInt(50); client.close();
        }
} }
```

```
import java.net.*;
import java.io.*;                                   // example TCP1 Client Program
public class NumberClient {
    public static void main(String args[]) throws IOException {
        System.out.println(" Before Connected");
        Socket server = new Socket("localhost",4446);
        System.out.println("Connected");
        DataInputStream input=null;
        try {
                input = new DataInputStream(server.getInputStream());
        }
        catch (IOException e) {  System.out.println(e);     }

        int aVal =input.readInt();
        System.out.println("Server said: "+aVal);
         server.close();
    }
}
```

```
import java.net.*;
import java.io.*;                          // example TCP2  ClientProgram
public class DateClient {

   public static void main (String args[]) throws IOException {

        Socket server = new Socket("localhost",1235);
        System.out.println("Connected");
        try {
                InputStream in = server.getInputStream();
        }
        catch (IOException e) {  System.out.println(e);     }
        byte b[ ] = new byte[100];

        int num = in.read(b);
        String date = new String(b);
        System.out.println("Server said: "+date); server.close();

   }
```

```
import java.net.*;
import java.io.*;
import java.util.*;                    // example TCP2 Server Program
public class DateServer {
        public static void main (String args[]) throws IOException {
        int port = 1235;
        ServerSocket server = new ServerSocket(port);
        while (true) {
                System.out.println("Waiting for client...");
                Socket client = server.accept();
                System.out.println("Client connected.");
                try{
                        OutputStream out = client.getOutputStream();
                }
                catch (IOException e) {  System.out.println(e);     }
                Date date = new Date();
                byte b[] = date.toString().getBytes();
                out.write(b); client.close();
} }
```

References



- Chapter 5
- In the book titled
- **Advanced Network Programming –**
- **Principles and Techniques**
  - Network Application Programming with Java
  - http://link.springer.com/book/10.1007%2F978-1-4471-5292-7