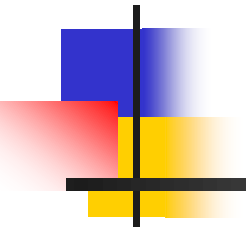
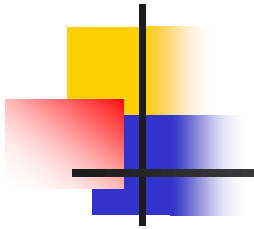


Network Programming

inter-process communication

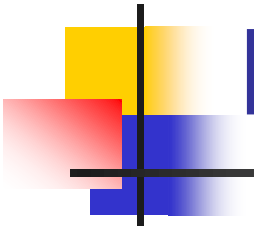
Sockets

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.



Multicast Sockets

A Multicast API should support the following primitive operations:

- (1) Join – allows a process to join a specific multicast group. A process may be a member of one or more multicast groups at the same time.
- (2) Leave – allows the process to stop participating in a multicast group.
- (3) Send – allows a process to send a message to all the processes of a multicast group.
- (4) Receive – allows a process to receive messages sent to a multicast group.

- **Note:** Of the five classes of available IP addresses the Class D range from 224.0.0.0 to 239.255.255.255 and is exclusively reserved for multicast groups.

Format and types of IP Address :

- The format of IP address is a **32-bit** numeric address written as four numbers separated by periods. Each number can be **zero to 255**. For example, **1.160.10.240** could be an IP address.

Class	Address Range	Supports
Class A	1.0.0.1 to 126.255.255.254	Supports 16 million hosts on each of 127 networks.
Class B	128.1.0.1 to 191.255.255.254	Supports 65,000 hosts on each of 16,000 networks.
Class C	192.0.1.1 to 223.255.254.254	Supports 254 hosts on each of 2 million networks.
Class D	224.0.0.0 to 239.255.255.255	Reserved for multicast groups.
Class E	240.0.0.0 to 254.255.255.254	Reserved for future use, or Research and Development Purposes.

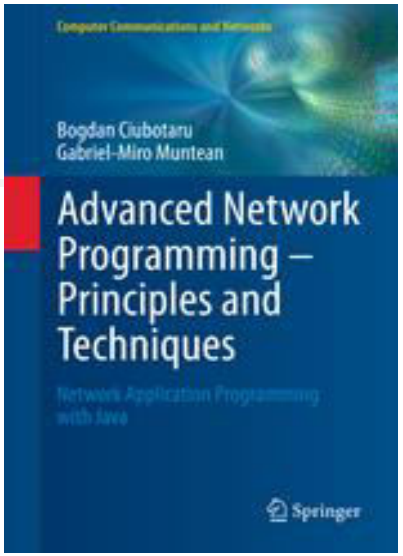
Client program 3

```
import java.io.*;
import java.net.*;
class multicastSender{
    public static void main(String[ ] args){
        try{
                InetAddress group = InetAddress.getByName("224.0.0.1“);
                MulticastSocket multicastSock = new MulticastSocket(3456);
                String msg = “Sending Network Programming  ?";
                DatagramPacket packet = new DatagramPacket(
                        msg.getBytes( ) , msg.length( ), group, 3456);
                multicastSock.send(packet);
                multicastSock.close( );
        }
        catch(Exception e){ }
    }
}
```

Server Program 3

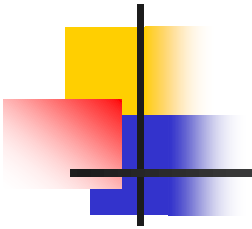
```
import java.io.*;
import java.net.*;
class multicastReceiver{
public static void main(String[ ] args){
        try{
                InetAddress group = InetAddress.getByName("224.0.0.1");
                MulticastSocket multicastSock = new MulticastSocket(3456);
                multicastSock.joinGroup(group);
                byte[ ] buffer = new byte[100];
                DatagramPacket packet = new DatagramPacket(buffer,
                buffer.length);
                multicastSock.receive(packet);
                System.out.println(new String(buffer));
                multicastSock.close( );
        }
        catch(Exception e){ }
}
}
```

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

- **DatagramSocket**
  - https://docs.oracle.com/javase/7/docs/api/java/net/DatagramSocket.html
- **MulticastSocket**
  - http://docs.oracle.com/javase/7/docs/api/java/net/MulticastSocket.html



- Project
  - For mobile computing students
    - viber or whatsup
  - For web technologies students
    - Firebase database