



Instituto Politécnico Nacional

**ESCOM “ESCUELA SUPERIOR DE
CÓMPUTO”**



DESARROLLO DE SISTEMAS DISTRIBUIDOS

TAREA 5: CHAT MULTICAST

PROFE: PINEDA GUERRERO CARLOS

ALUMNO: Rojas Alvarado Luis Enrique

BOLETA: 2014010995

GRUPO: 4CM5

DESARROLLO:

Para ésta práctica se necesita implementar un chat multicast con las funciones envia_mensaje y recibe_mensaje ya facilitadas por el profesor en clases pasadas, solo tenemos que completar la clase worker, donde recibiremos los mensajes, por lo que crearemos las instancias para crear el grupo y el socket multicast, así como la string de 100 caracteres para capturar de la mejor manera un mensaje. Y en el método main, en un ciclo infinito enviaremos el mensaje apoyandonos del método readline() en el grupo y socket ya definido anteriormente

```
static class Worker extends Thread{

    public void run() {
        // En un ciclo infinito se recibirán los mensajes enviados al grupo
        // 230.0.0.0 a través del puerto 50000 y se desplegarán en la pantalla
        for (;;) {
            try {

                /**Para obtener el grupo invocamos el método getByName() de la clase InetAddress:
                */
                InetAddress grupo = InetAddress.getByName("230.0.0.0");

                /**Luego obtenemos un socket asociado al puerto 50000, creando una instancia de la clase MulticastSocket: */
                MulticastSocket socket = new MulticastSocket(50000);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
/**Para que el cliente pueda recibir los mensajes enviados al grupo 230.0.0.0 unimos el socket al grupo
* utilizando el método joinGroup() de la clase MulticastSocket:
*/
socket.joinGroup(grupo);
/**Entonces el cliente puede recibir los mensajes enviados al grupo por el servidor.
*/
// System.out.println("Esperando datagrama a...");
/* recibe una string
byte[] a = recibe_mensaje(socket, 100);
System.out.println(new String(a, "UTF-8"));

socket.leaveGroup(grupo);
socket.close();

} catch (Exception e) {
    e.printStackTrace();
}

// fin
}run()
}
```

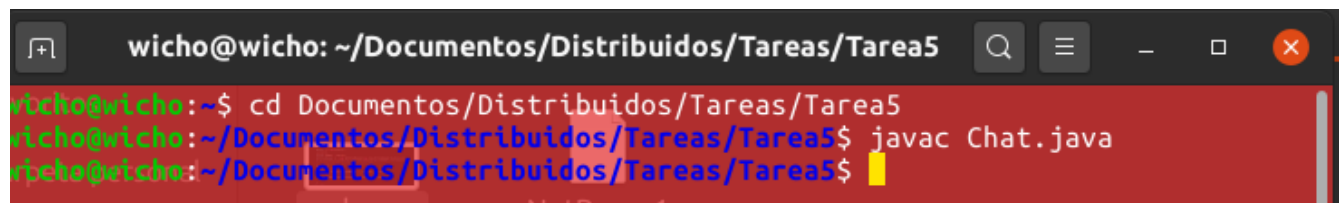
```

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

    Worker w = new Worker();
    w.start();
    String nombre = args[0];
    BufferedReader b = new BufferedReader(new InputStreamReader(System.in));
    // En un ciclo infinito se leerá los mensajes del teclado y se enviará
    // al grupo 230.0.0.0 a través del puerto 5000
    for (;;) {
        String msg =
        String salida = b.readLine() + nombre + " escribe: " + msg;
        /* Enviando un string */
        envia_mensaje(salida.getBytes(), "230.0.0.0", 50000);
    },
    }
    // Fin main
}

```

Recuperamos en la línea de comandos el nombre y lo adjuntamos al mensaje, posteriormente invocamos al método `envia_mensaje` y quedará listo. Compilando el programa:

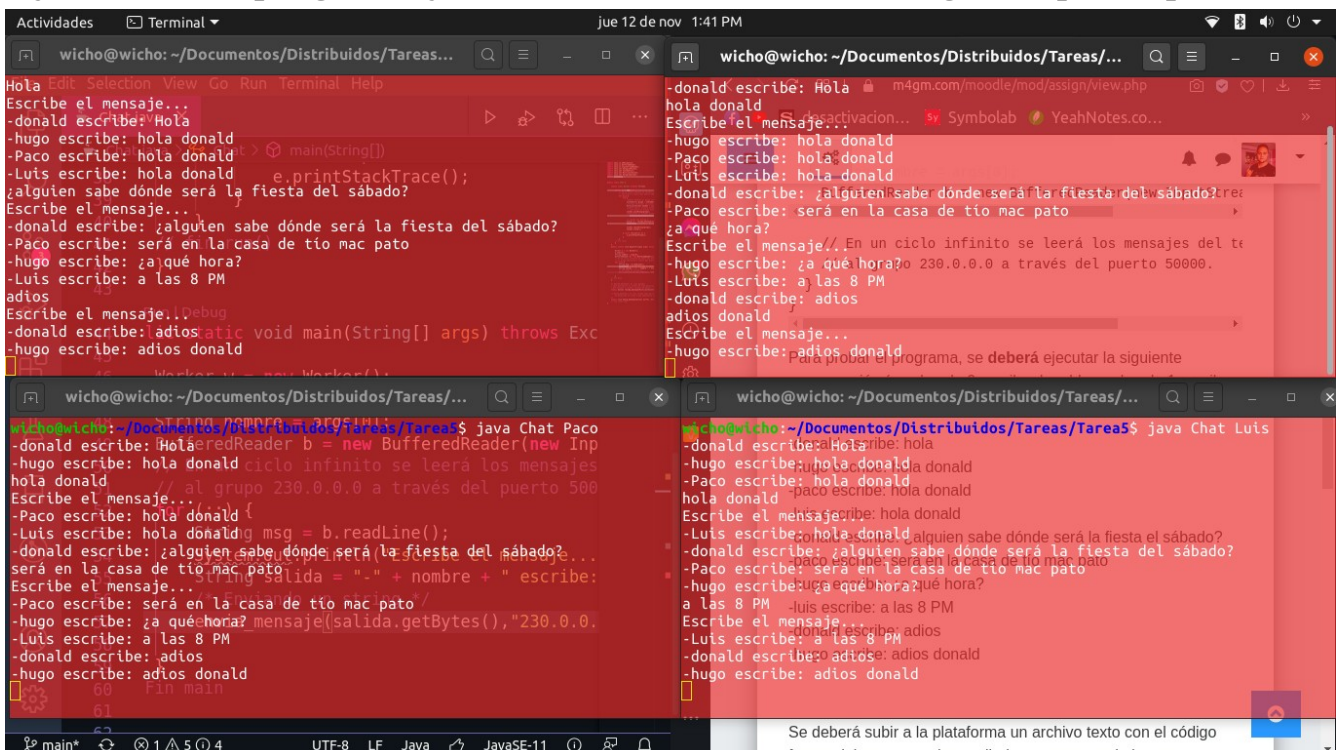


```

wicho@wicho: ~/Documentos/Distribuidos/Tareas/Tarea5
wicho@wicho:~/Documentos/Distribuidos/Tareas/Tarea5$ javac Chat.java
wicho@wicho:~/Documentos/Distribuidos/Tareas/Tarea5$

```

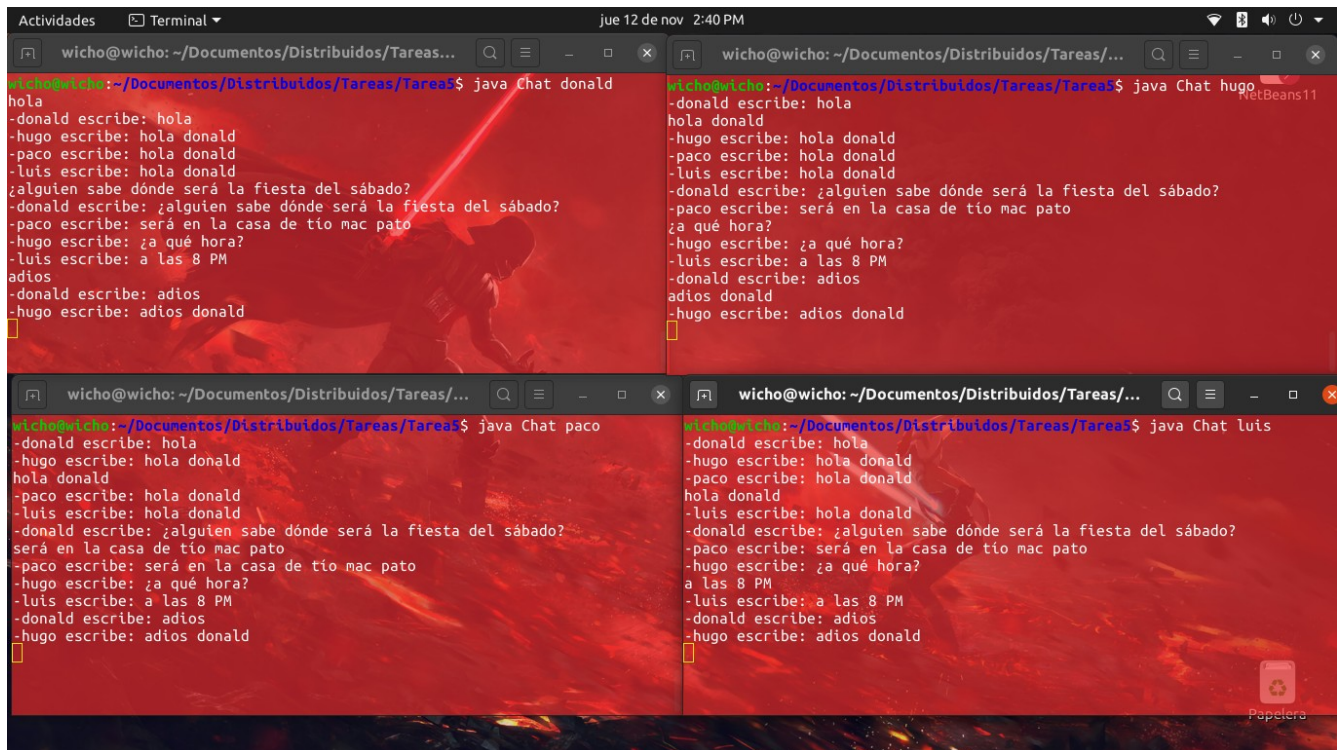
Ejecutando el programa y haciendo la conversación sugerida por el profesor:



```

wicho@wicho: ~/Documentos/Distribuidos/Tareas/Tarea5$ java Chat Paco
-donald escribe: Hola
-hola donald
Escribe el mensaje...
-donald escribe: Hola
-hugo escribe: hola donald
-Paco escribe: hola donald
-Luis escribe: hola donald
¿alguien sabe dónde será la fiesta del sábado?
Escribe el mensaje...
-donald escribe: ¿alguien sabe dónde será la fiesta del sábado?
-Paco escribe: será en la casa de tío mac pato
-hugo escribe: ¿a qué hora?
-Luis escribe: a las 8 PM
adios
Escribe el mensaje...
-donald escribe: adios
-hugo escribe: adios donald
wicho@wicho: ~/Documentos/Distribuidos/Tareas/Tarea5$ java Chat Luis
-donald escribe: Hola
-hola donald
Escribe el mensaje...
-donald escribe: Hola
-hugo escribe: hola donald
-Paco escribe: hola donald
-hola donald -paco escribe: hola donald
Escribe el mensaje...
-donald escribe: ¿alguien sabe dónde será la fiesta del sábado?
-Paco escribe: será en la casa de tío mac pato
-hugo escribe: ¿a qué hora?
a las 8 PM -Luis escribe: a las 8 PM
Escribe el mensaje...
-Luis escribe: adios
-donald escribe: adios donald
-hugo escribe: adios donald

```



The image shows four terminal windows arranged in a 2x2 grid, each displaying the output of a Java chat application. The windows are titled 'wicho@wicho: ~/Documentos/Distribuidos/Tareas/...' and show the following chat logs:

- Top Left (Client: donald):**
hola
-donald escribe: hola
-hugo escribe: hola donald
-paco escribe: hola donald
-luis escribe: hola donald
¿alguien sabe dónde será la fiesta del sábado?
-donald escribe: ¿alguien sabe dónde será la fiesta del sábado?
-paco escribe: será en la casa de tío mac pato
-hugo escribe: ¿a qué hora?
-luis escribe: a las 8 PM
adios
-donald escribe: adios
-hugo escribe: adios donald
- Top Right (Client: hugo):**
hola donald
-donald escribe: hola
-hugo escribe: hola donald
-paco escribe: hola donald
-luis escribe: hola donald
-donald escribe: ¿alguien sabe dónde será la fiesta del sábado?
-paco escribe: será en la casa de tío mac pato
¿a qué hora?
-hugo escribe: ¿a qué hora?
-luis escribe: a las 8 PM
-donald escribe: adios
adios donald
-hugo escribe: adios donald
- Bottom Left (Client: paco):**
hola
-donald escribe: hola
-hugo escribe: hola donald
hola donald
-paco escribe: hola donald
-luis escribe: hola donald
-donald escribe: ¿alguien sabe dónde será la fiesta del sábado?
será en la casa de tío mac pato
-paco escribe: será en la casa de tío mac pato
-hugo escribe: ¿a qué hora?
-luis escribe: a las 8 PM
-donald escribe: adios
-hugo escribe: adios donald
- Bottom Right (Client: luis):**
hola donald
-donald escribe: hola
-hugo escribe: hola donald
-paco escribe: hola donald
hola donald
-luis escribe: hola donald
-donald escribe: ¿alguien sabe dónde será la fiesta del sábado?
-paco escribe: será en la casa de tío mac pato
-hugo escribe: ¿a qué hora?
a las 8 PM
-luis escribe: a las 8 PM
-donald escribe: adios
-hugo escribe: adios donald

CONCLUSIONES

Para ésta práctica se pudo observar el comportamiento distribuido de un programa de chat multicast, en el cual, a diferencia del unicast no se tiene que cuidar la linealidad de la transmisión de mensajes, ya que estando dentro del grupo, es más fácil que se comuniquen sin tener una secuencialidad, respetando un orden, aquí todos reciben el dato (en este caso mensaje/cadena/string), que en unicast, un servidor (o un nodo en específico) se encargaba de “almacenar” los datos que se enviaban y trabajar con un hilo cada conexión o cliente, además de que tiene un especial cuidado con el paralelismo o concurrencia.