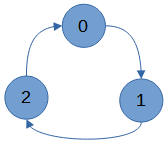
**Tarea 4. Implementación de un token-ring**

Desarrollar un programa en Java, el cual implementará un token que pasará de un nodo a otro nodo, en una topología lógica de anillo. El anillo constará de tres nodos:



El token será un número entero de 32 bits. El nodo 0 inicializará el token con 1.

El nodo 0 enviará el token al nodo 1, entonces el nodo 1 recibirá el token y lo enviará al nodo 2. El nodo 2 recibirá el token y lo enviará al nodo 0.

Cada nodo contará y desplegará las veces que recibe el token. Cuando la cuenta en el nodo 0 llegue a 1000, el nodo 0 deberá terminar su ejecución, entonces ¿qué pasará con los otros nodos?

Consideremos el siguiente programa (notar que este programa es un cliente y un servidor):

class Token

{

static DataInputStream entrada;

static DataOutputStream salida;  
 static boolean primera\_vez = true;  
 static String ip;  
 static int nodo;

static int token;  
 static int contador = 0;

static class Worker extends Thread

{

public void run()

{

**Algoritmo 1**  
 }

}

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

{

if (args.length != 2)

{

System.err.println("Se debe pasar como parametros el numero del nodo y la IP del siguiente nodo");

System.exit(1);

}

nodo = Integer.valueOf(args[0]); // el primer parametro es el numero de nodo  
 ip = args[1]; // el segundo parametro es la IP del siguiente nodo en el anillo

**Algoritmo 2**

}

}

Se deberá implementar los siguientes algoritmos:

**Algoritmo 1**

1. Declarar la variable **servidor** de tipo ServerSocket,

2. Asignar a la variable **servidor** el objeto: new ServerSocket(50000)

3. Declarar la variable **conexion** de tipo ﻿Socket.

4. Asignar a la variable **conexion** el objeto servidor.accept().

5. Asignar a la variable **entrada** el objeto new DataInputStream(conexion.getInputStream()).

**Algoritmo 2**

1. Declarar la variable **w** de tipo Worker.

2. Asignar a la variable **w** el objeto new Worker().

3. Invocar el método w.start().

4. Declarar la variable **conexion** de tipo Socket. Asignar null a la variable **conexion**.

5. En un ciclo:

5.1 En un bloque try:

5.1.1 Asignar a la variable **conexion** el objeto Socket(ip,50000).

5.1.2 Ejecutar break para salir del ciclo.

5.2 En el bloque catch:

5.2.1 Invocar el método Thread.sleep(500).

6. Asignar a la variable **salida** el objeto new DataOutputStream(conexion.getOutputStream()).

7. Invocar el método w.join().

8. En un ciclo:

8.1 Si la variable **nodo** es cero:

8.1.1 Si la variable **primera\_vez** es true:

8.1.1.1 Asignar false a la variable **primera\_vez**.

8.1.1.2 Asignar 1 a la variable **token**.

8.1.2 Si la variable **primera\_vez** es false:

8.1.2.1 Asignar a la variable **token** el resultado del método entrada.readInt().

8.1.2.2 Incrementar la variable **contador**.

8.1.2.3 Desplegar las variables **nodo**, **contador** y **token**.

8.2 Si la variable **nodo** no es cero:

8.2.1 Asignar a la variable **token** el resultado del método entrada.readInt().

8.2.2 Incrementar la variable **contador**.

8.2.3 Desplegar las variables **nodo**, **contador** y **token**.

8.3 Si la variable **nodo** es cero y la variable **contador** es igual a 1000:

8.3.1 Salir del ciclo.

8.6 Invocar el método salida.writeInt(**token**).

Notar que el algoritmo 1 debe implementarse dentro de un bloque try.

Para cada nodo se deberá crear una **máquina virtual Ubuntu en Azure** con 1 CPU, 1 GB de RAM y disco HDD estándar.

El nombre de cada máquina virtual deberá ser el número de boleta del alumno, un guión y el número de nodo, por ejemplo, si el número de boleta del alumno es 12345678, entonces el nodo 0 deberá llamarse: 12345678-0, el nodo 1 deberá llamarse 12345678-1, y así sucesivamente. **No se admitirá la tarea** si los nodos no se nombran como se indicó anteriormente.

Recuerden que deben **eliminar las máquinas virtuales** cuando no las usen, con la finalidad de ahorrar el saldo de sus cuentas de Azure.

Para que las máquinas virtuales puedan recibir conexiones a través del puerto 50000, se deberá abrir este puerto en cada máquina virtual.

Para abrir el puerto 50000 en una máquina virtual:

1. Entrar al portal de Azure
2. Seleccionar "Maquinas virtuales".
3. Seleccionar la máquina virtual.
4. Dar click en "Redes".
5. Dar click en el botón "Agregar regla de puerto de entrada".
6. En el campo "Intervalos de puertos de destino" ingresar: 50000
7. Seleccionar el protocolo: TCP
8. En el campo "Nombre" ingrear: Puerto\_50000

Se deberá subir a la plataforma un archivo PDF que incluya: portada, **captura de pantalla de la creación de las máquinas virtuales**, captura de pantalla de la compilación y ejecución del programa en cada máquina virtual (capturar las pantallas cuando termina el programa en cada nodo), el código fuente del programa desarrollado (como texto no como imágen) y conclusiones. **No se admitirá la tarea** si no incluye la captura de pantalla de la creación de las máquinas virtuales.

Se deberá incluir la imagen completa de cada captura de pantalla, **no se admitirá la tarea** si la captura de las pantallas no es completa, es decir, si sólo se muestra un corte de la pantalla.

Las capturas de pantalla deberán ser legibles suponiendo que se requieran imprimier en papel, es decir, no se deberá reducir las imágenes correspondientes a las capturas de pantalla hasta un punto en el que no se pueda leer su contenido si se imprimen en papel (normalmente una captura de pantalla completa legible debería ocupar media página).

Valor de la tarea: 20% (1.2 puntos de la primera evaluación parcial)