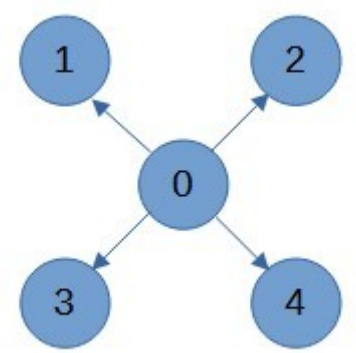
Planteamiento del problema

En esta tarea vamos a desarrollar un programa distribuido,el cual calculará una aproximación de PI utilizando la serie de Gregory-Leibniz.

La serie tiene la siguiente forma: 4/1-4/3+4/5-4/7+4/9-4/11+4/13-4/15 ...

El programa se va ejecutar en forma distribuida sobre cinco nodos.

Se implementará la siguiente topología lógica en estrella:



El nodo 0 actuará como cliente y los nodos 1, 2, 3 y 4 actuarán como servidores.

El nodo 0 deberá implementar re-intentos de conexión a cada servidor, de manera que se pueda iniciar la ejecución de cada nodo en cualquier orden.

Se debe desarrollar un solo programa por tanto será necesario pasar como parámetro al programa de nodo actual, de manera que el programa pueda actuar como servidor o como cliente, según el número de nodo que pasa como parámetro.

Debido que los nodos van a ejecutar en la misma computadora, cada nodo servidor deberá abrir un puerto diferente, por ejemplo,el nodo 1 podría abrir elpuerto 50001,el nodo 2 podría abrir elpuerto 50002,el nodo 3 podría abrir elpuerto 50003 y elnodo 4 podría abrir el puerto 50004.

Desarrollo de la tarea

A continuación explicaremos cada uno de los métodos de nuestra clase Nodo, el cual puede funcionar como servidor o como cliente.

Empezamos con las variables globales y su función para el programa:

- ServerSocket servidor: Será el socket para iniciar el servidor del nodo 1,2,3 y 4.
- Socket Cliente: Es el socket del cliente para establecer conexión con los servidores.
- DataOutputStream salida :Es el canalde salida para que los servidores puedan enviar el resultado de su sumatoria al cliente.
- double Nodos[] :Este arreglo de longitud 4 lo usaremos para guardar la sumatoria de cada servidor una vez que sea enviada al cliente.
- DataInputStream entrada:Es el canal de entrada para que nuestro cliente pueda recibir las sumatorias por parte de los servidores.

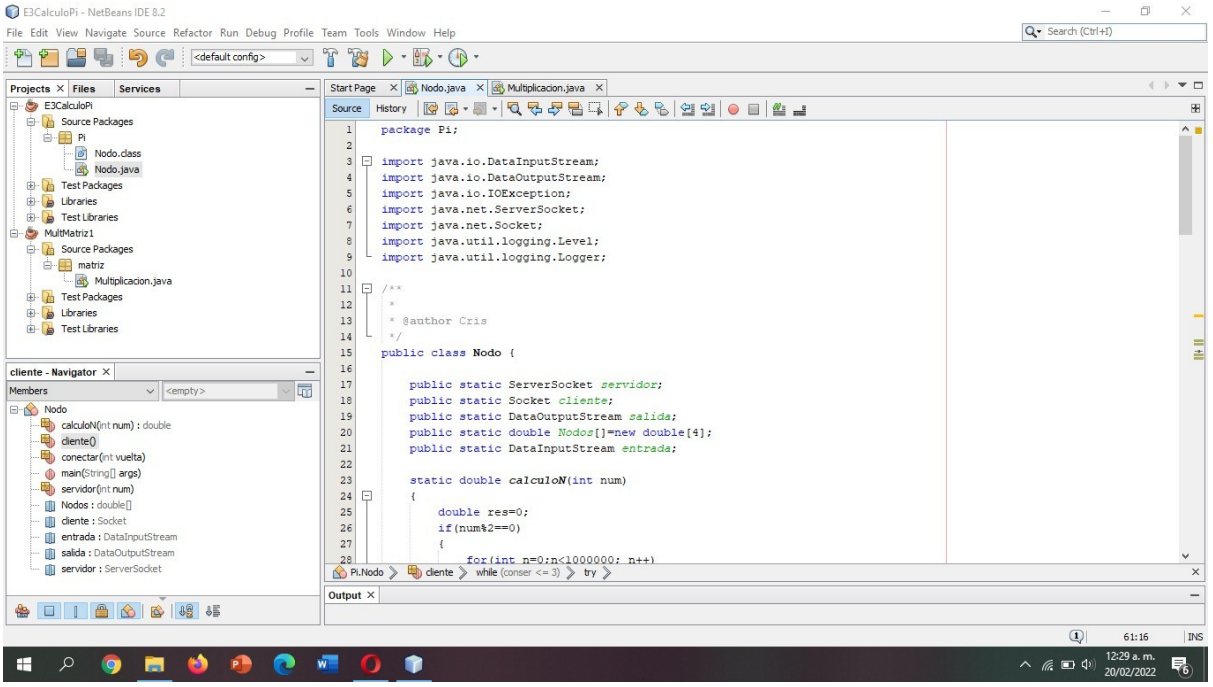


Imagen 1: Variables globales de nuestra clase Nodo

Al ejecutar el programa debemospasar como argumento elnúmero de nodo, para determinar sise trata de un cliente (0) o un servidor (1,2,3 o 4), en caso de que no se pase un argumento o un número de nodo válido se mostrará un mensaje explicando el error cometido.En el caso de que el argumento sea un 0 se ejecuta el metodo cliente() donde se llevará a cabo toda la logística para que el cliente establezca conexión con los servidores y reciba su sumatoria, y en caso de tratarse de un servidor se ejecuta el método servidor() donde se llevará a cabo toda la logística para que cada servidor inicie, se conecten con el cliente y envíe la sumatoria a dicho cliente.

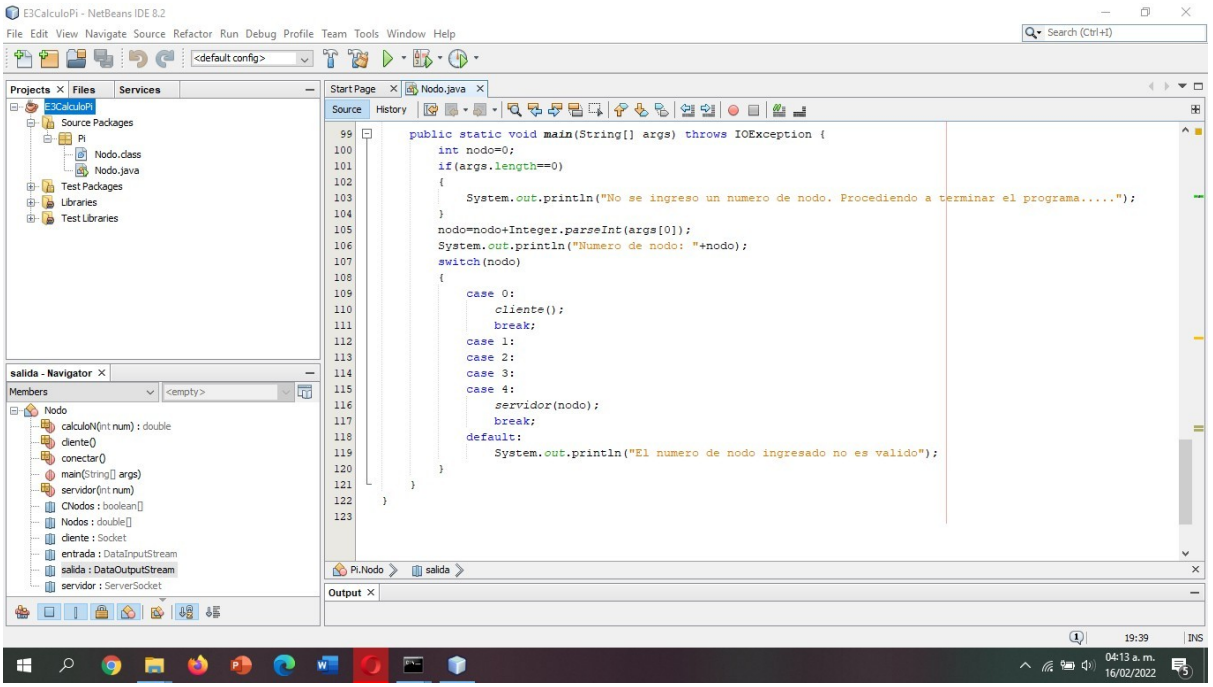


Imagen 2: Código inicial donde se determina si se trata de un nodo servidor o un nodo cliente

Métodos del servidor

Ahora veremos todos los métodos que usamos para que los servidores realicen la conexión con el cliente y el envío de la sumatoria.

Método servidor(int num)

Usaremos nuestro ServerSocket para iniciar nuestro servidor, pero debido a que hay 4 servidores que deben establecer con el cliente, para poder diferenciar entre ellos usaremos un número de puerto específico para cada uno, el cual se trata del puerto 50000 mas el numero de nodo que corresponde al servidor que deseamos inicializar.
Posteriormente esperamos a que el cliente establezca conexión con nuestro servidor llamando al método accept de la clase ServerSocket. Una vez establecida la conexión con el cliente procedemos a enviar la sumatoria, la cual es un dato de tipo double, por lo que usaremos el método writeDouble de nuestro canal de salida. Como podemos ver el dato que enviamos es la sumatoria que nos regresa el método calculoN y posterior a esto procedemos a cerrar la conexión con el cliente.

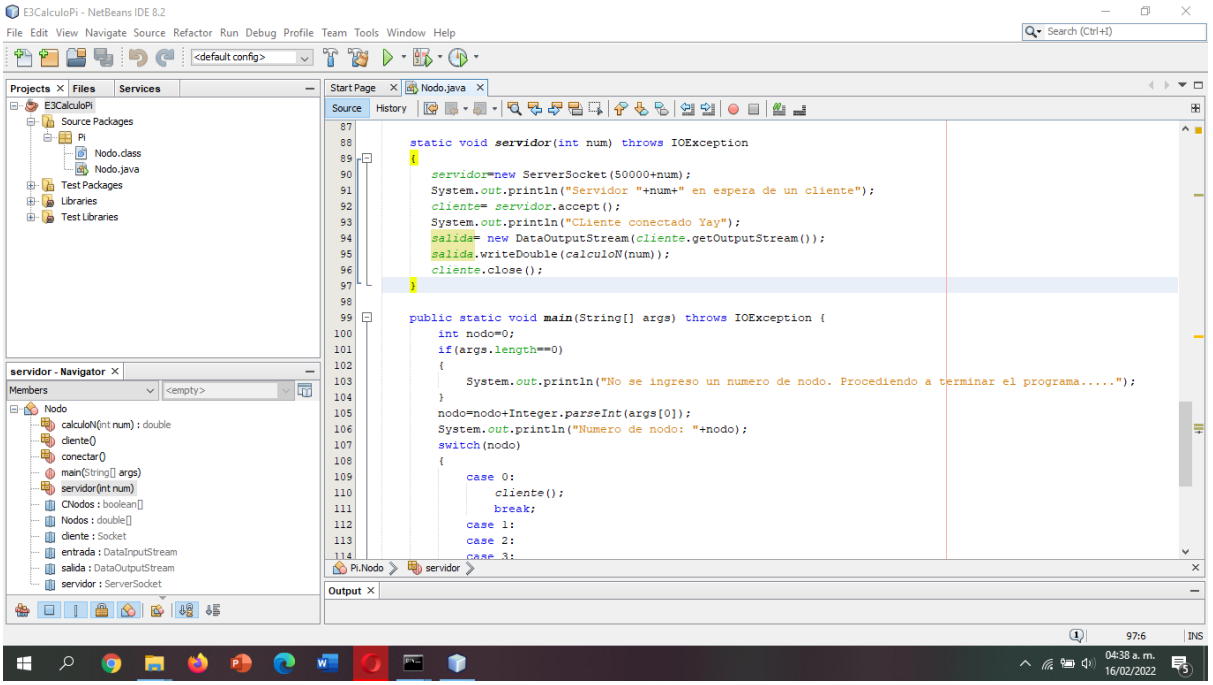


Imagen 3 : Método servidor donde se establece la conexión con el cliente y se envia la sumatoria

Método calculoN(int num)

Este método es donde se lleva a cabo el cálculo de la sumatoria, para el cual es necesario el número de nodo (servidor) con el fin de determinar qué tipo de sumatoria realizar, así para sustituir el número de nodo en la fórmula de la sumatoria. Una vez terminada la sumatoria de 1000000 de términos, procedemos a retornar el resultado.

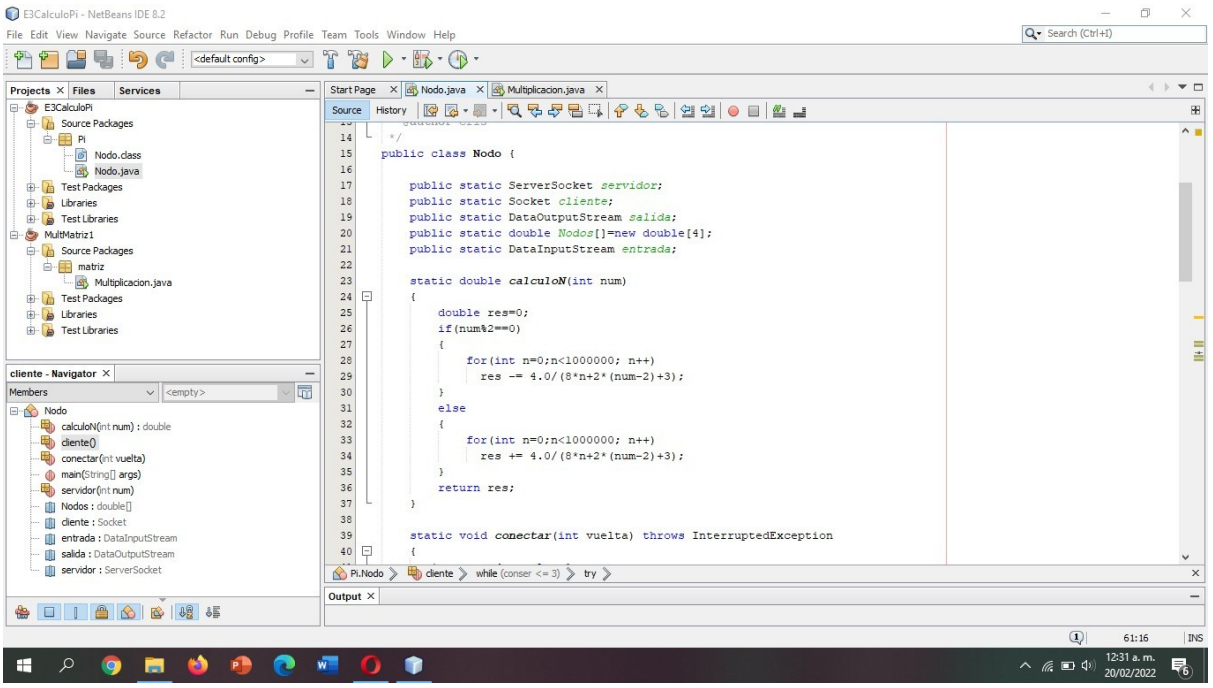


Imagen 4: Método calculoN donde se calcula la sumatoria correspondiente al número de nodo

Metodos del cliente

Ahora veremos los métodos para que el cliente pueda establecer con los servidores, recibir las sumatorias de cada servidor y estar constantemente tratando de conectarse con los servidores en caso de que alguno no haya iniciado.

Metodo cliente()

El método cliente hace uso de una variable de tipo entero que usaremos como contador que aumentará en 1 cada vez que el cliente se logre conectarcon un servidor. Esto lo llevaremos a cabo con ayuda deun ciclo while dentro delcual mandamos a llamar al método conectar, al cual le pasamos como parámetro nuestro contador para que el cliente sepa con que servidor tiene que establecer conexión. Como vimos anteriormente en el metodo conectar, el cliente se mantendrá en bucle hasta lograr conexión con el servidor, por lo que una vez que se logre la conexión, se reciba la sumatoria que envía el servidor al cliente y se cierre la conexión entre el cliente y el servidor (Es decir los pasos que se llevan a cabo en elmetodo conectar cuando la conexión fue un éxito) aumentamos en 1 nuestro contador y procedemos a verificar si se cumple la condición del while, el cual establece que si el contador es menor o igual a 3 (Es decir que todavía falta establecer conexión con uno o más servidores), seguiremos llamando al método conectar hasta haber logrado la conexión con los cuatro servidores. Posterior a esto y para finalizar mostramos mediante un mensaje la sumatoria de los valores mandados por cada servidor los cuales guardamos en cada uno de los índices de nuestro arreglo de tipo double “Nodos”. Dicha sumatoria nos da el valor de PI de acuerdo a la serie de Gregory-Leibniz.

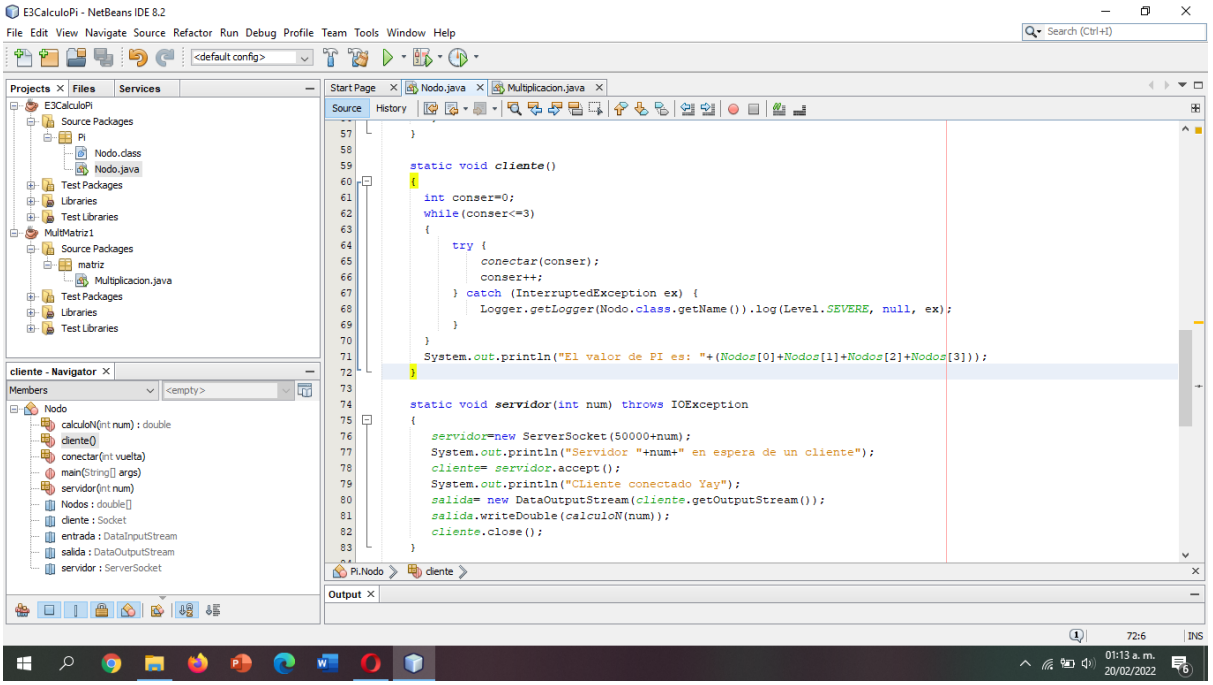


Imagen 5 . Método conectar donde ciclamos al cliente hasta lograr la conexión con los 4 servidores.

Metodo conectar(int vuelta)

Este método es elque lleva a cabo los reintentos de conexión, para elcual necesitamos recibir como parámetro un valor de tipo entero el cual se trata del valor actual del contador, pero debido a que el valor inicial del contador es 0 será necesario crear una variable la cual sea el aumento en 1 del valor actual del contador permitiendo conectarnos con el servidor 1 hasta el4, ya que sirecordamos en el método cliente el ciclo while mandara a llamar a este método hasta que el valor del contador sea mayor o igual a 4, haciendo que el rango de valores del contador sea del 0 al 3, pero nosotros al necesitar conectarnos del servidor 1 al 4 tendremos que aumentar en 1 el valor del contador, haciendo que el cliente se conecte en orden con cada servidorempezando con el servidor 1 y terminando con el servidor 4. Como podemos ver usamos la cláusula try-catch para el intento de conexión con el servidor,cuyo número de puerto es la suma del puerto 50000 y el número del servidor. En caso de que el servidor no haya sido ejecutado y no se encuentre en espera de un cliente, se produce una excepción, por lo que modificamos elbloque catch para poner a dormir a nuestro cliente por 3 segundos antes de que vuelva a intentar con el servidor. Mientras no se logre realizar la conexión con el servidor,nuestro cliente se encontrará en un ciclo for infinito del cual solo podrá salir en cuando logre conectarse con el servidor.Una vez lograda conexión con el servidor creamos el canal de entrada para recibir la sumatoria que el servidor manda alcliente,haciendo uso delmétodo readDouble de nuestro canal de entrada y con ayuda de nuestro arreglo de tipo double guardamos la sumatoria en el índice correspondiente de acuerdo al servidor con el que estamos realizando la conexión; en este caso usamos elparámetro que recibe el método para indicar en qué índice del arreglo se

guardará la sumatoria. Por último procedemos a cerrar la conexión entre el cliente y el servidor y ejecutamos un break para salir del ciclo infinito ya que el cliente al fin logró conectarse al servidor.

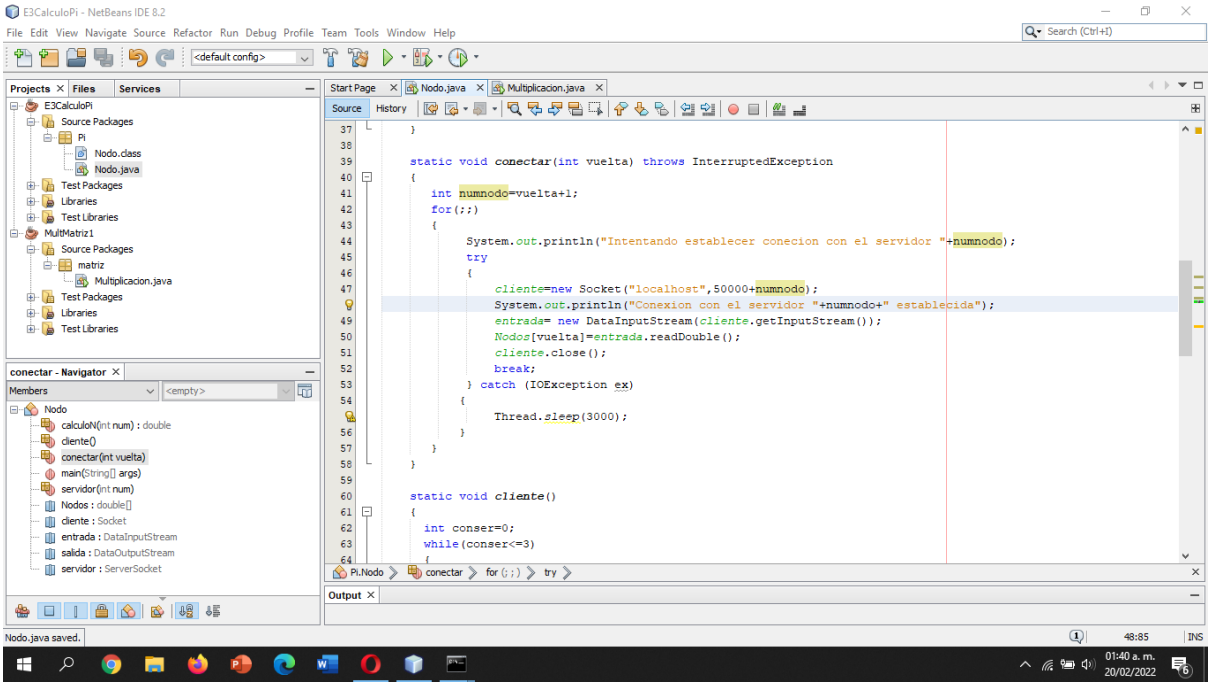


Imagen 6: Metodo conectar donde se hacen reintentos de conexión hasta establecer conexión con el servidor

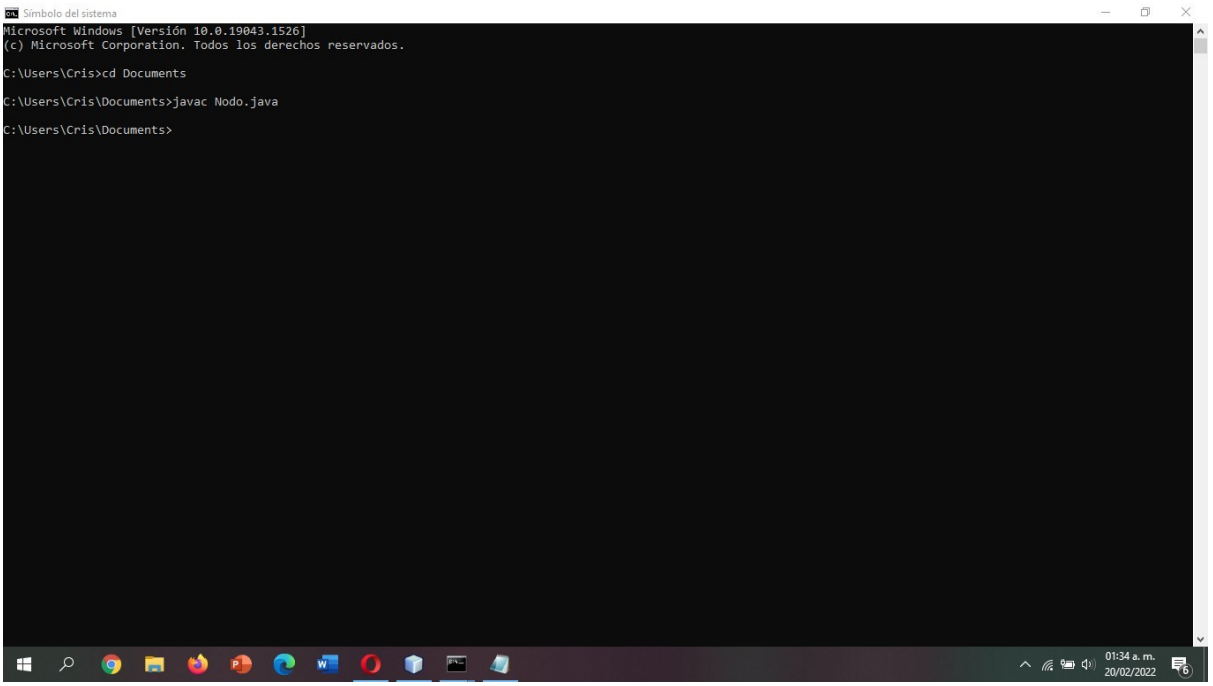


Imagen 7: Compilación del programa

Pruebas de funcionamiento del programa

Como podemos observar en la siguiente imagen tenemos a todos los servidores en espera de conectarse con el cliente por lo que ahora procederemos a ejecutar a nuestro cliente y veremos cómo de poco en poco se va conectando a cada uno de ellos sin ningún problema y procederá a desplegar el valor de PI.

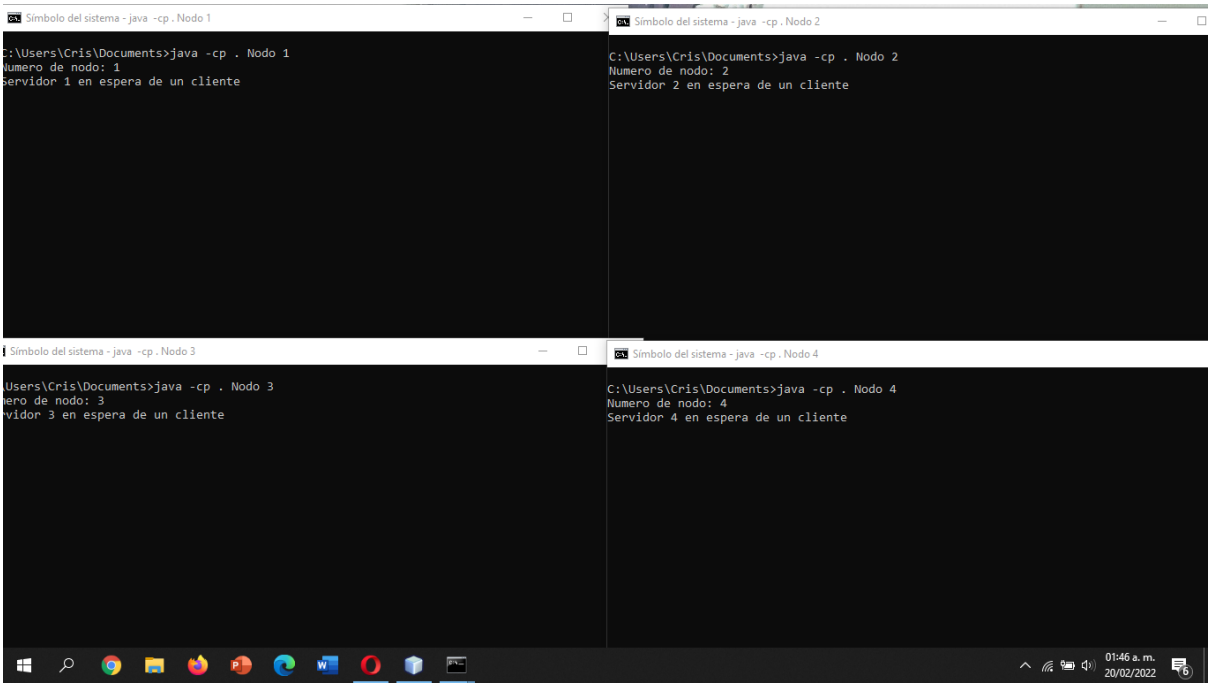


Imagen 8: Ejecución de los 4 servidores

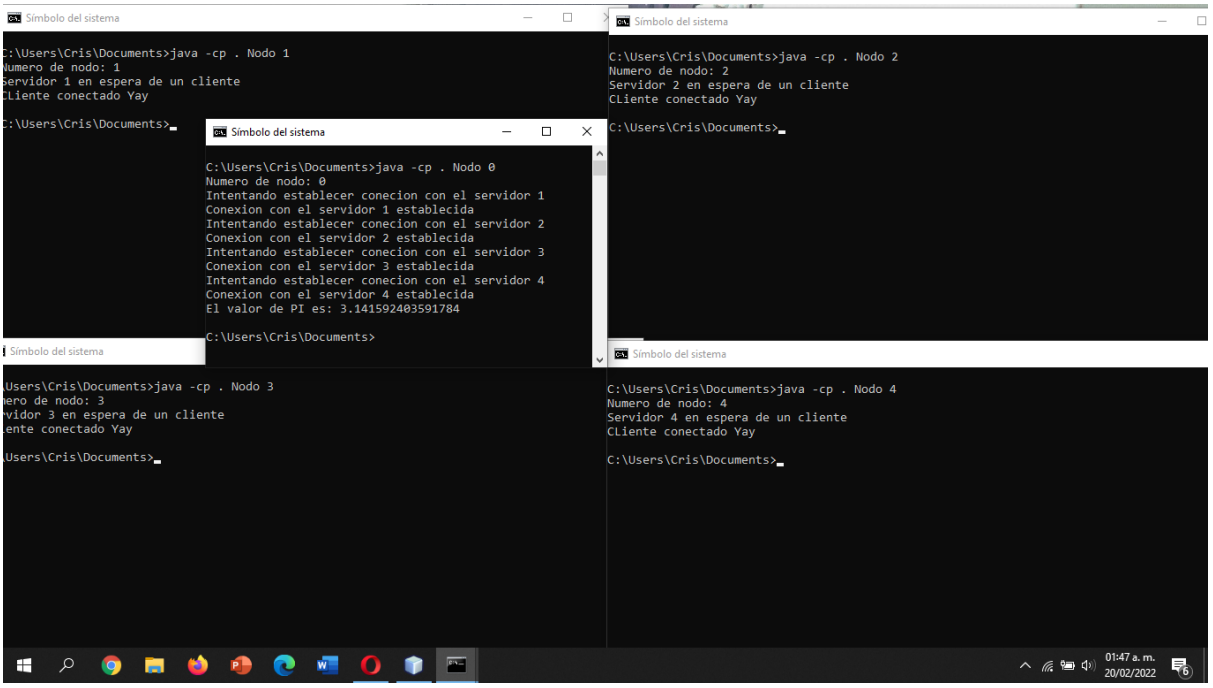


Imagen 9 : Ejecución del cliente y su conexión con cada servidor

En esta prueba no hay mayor problema ya que al estar los 4 servidores en línea, el cliente se podrá conectar a cada uno de ellos en cuestión de segundos.

Pero si ahora solo ejecutamos el servidor 2 y 4, el cliente en un inicio se verá ciclado tratando de conectarse al servidor 1, una vez que logre conectarse al servidor 1 no tendrá problemas para conectarse al servidor 2 ya que este ya se encuentra en espera a que el cliente se conecte. Algo similar ocurrirá con el servidor 3 y 4, que mientras no se logre conectar al servidor 3, el cliente estará reintentando conectarse cada 3 segundos y una vez que se logre conectar al servidor 3, la conexión con el servidor 4 no será problema ya que este se encuentra en espera a que el cliente establezca conexión con el.

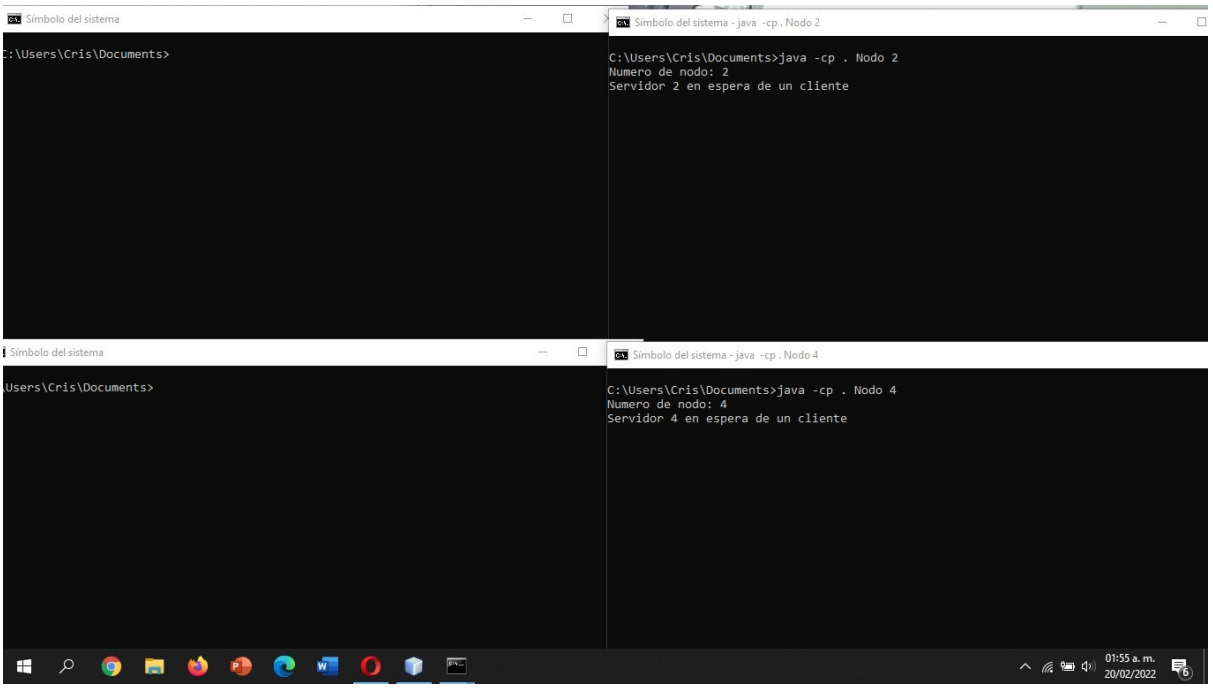


Imagen 10: Ejecución únicamente del servidor 2 y 4

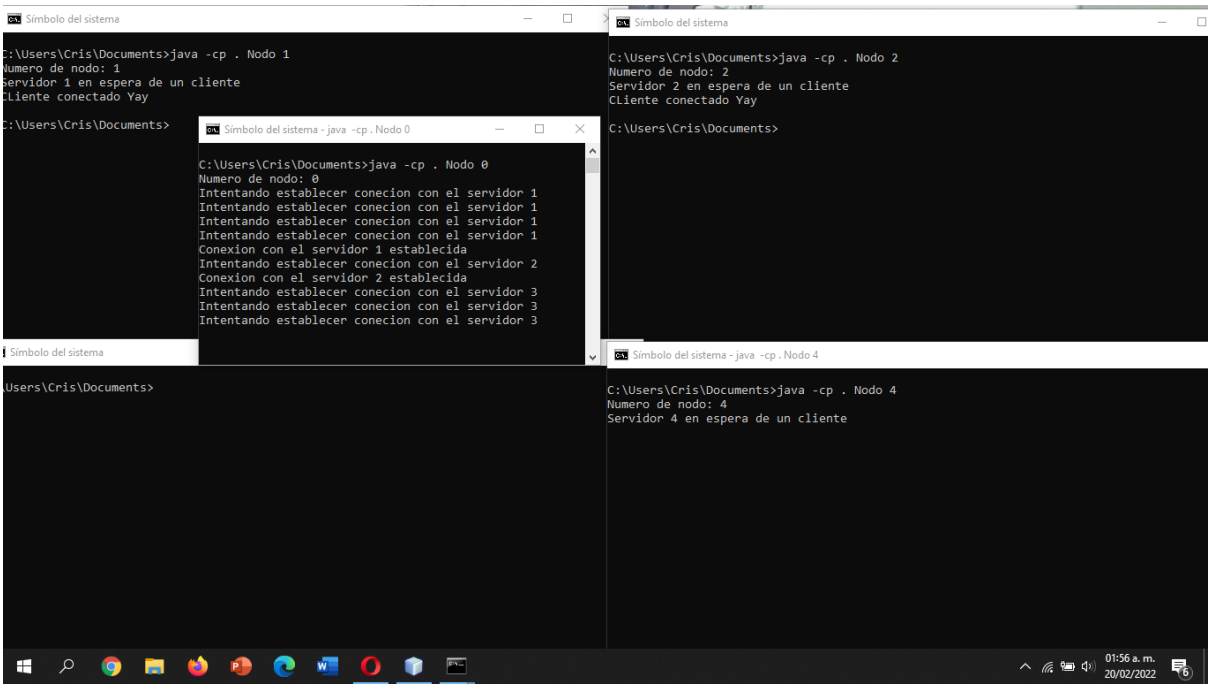


Imagen 11: El cliente logra conectar con el servidor 2 después de 9 segundos tratando de conectarse al servidor 1, ya que es el tiempo que pasó hasta se ejecutó el servidor 1 y el cliente logra establecer conexión a partir del 4 intento.

Como podemos observar el cliente hace varios intentos por conectarse al servidor 1, lo cual logra al intento número 4,lo que significa que el servidor1 lo ejecutamos en los 3 segundos de espera entre el intento 3 y el 4, ya que al cuarto intento es cuando el cliente logra establecerla conexión con elservidor 1,debido a que este ya se encontraba en espera a la conexión con el cliente. Y en el caso del servidor 2 este será exitoso al primer intento,debido a que estaba en espera desde el momento que ejecutamos a nuestro cliente y una vez que se logre conectar alservidor 1,no habrá problemas para conectarse al servidor 2.

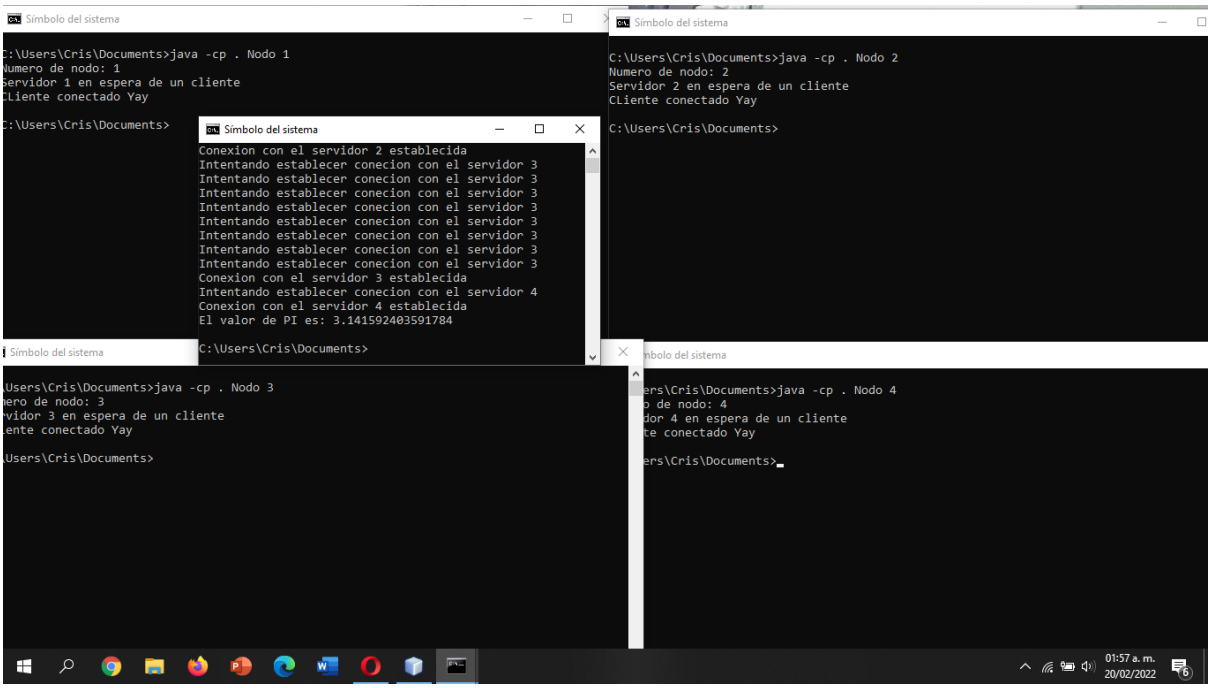


Imagen 12: El cliente logra conectar con el servidor 4 después de 21 segundos intentando conectarse al servidor 3, ya que es el tiempo que pasó hasta que se ejecutó el servidor 3 y el cliente logra establecer conexión a partir del intento 8.

Similar a lo ocurrido con el servidor1, en este caso dejamos pasar un tiempo de 27 segundos,donde en los últimos tres segundos es donde se ejecuta el servidor 3 y se pone en espera de conexión con el cliente,para que en el intento número 8 la conexión con el servidor 3 por fin sea exitosa. Igualmente que con el servidor 2, el cliente no tendrá ningún problema para establecer conexión con el servidor 4,ya que este estuvo esperando al cliente para establecer conexión, desde el momento que ejecutamos al cliente.

Al final podemos ver que gracias a los reintentos de conexión podemos ejecutar un cliente sin necesidad de que el servidor haya iniciado primero, ya que habrá momentos en los que es necesario que el cliente sea ejecutado primero o tenga que esperar a un servidor para establecer conexión. En este caso cuando falla alintentar conectar al servidor 1 y 3,el cliente no termina la ejecución del método,sino que sigue intentando hasta lograr conectarse, y una vez que logra conectarse a cada servidor procede a mostrar el valor de PI.