

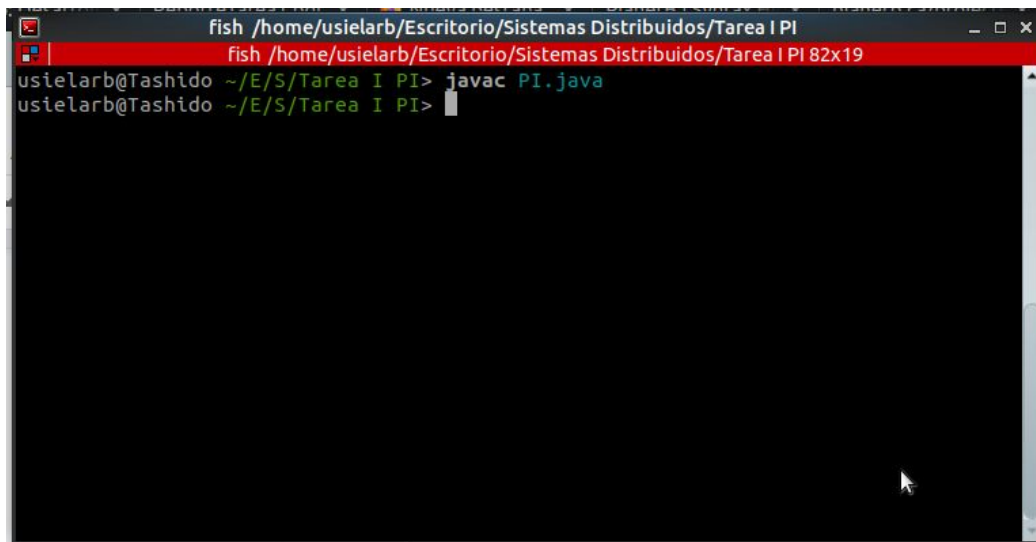


Alumno: Arista Bojorges Victor Usiel
Grupo: 4CV1

Instrucciones: Se deberá subir a la plataforma un archivo ZIP que contenga el código fuente del programa desarrollado y un documento PDF con la captura de pantalla de la compilación y ejecución del programa. El archivo PDF deberá incluir una descripción de cada captura de pantalla.

Capturas:

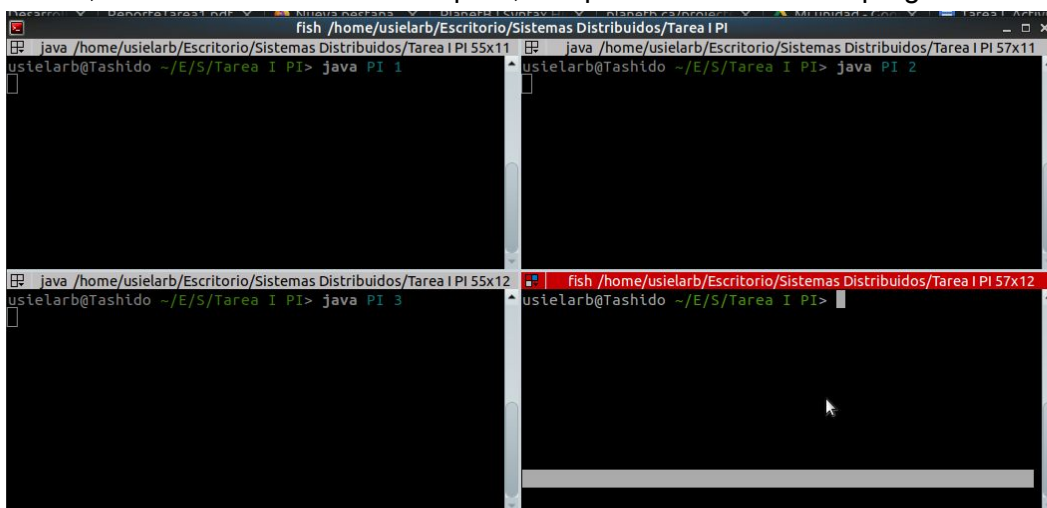
En la *Figura I* se muestra como se realiza la compilación del programa mostrando que no existe ningún error.



```
fish /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI
fish /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI 82x19
usiellarb@Tashido ~/E/S/Tarea I PI> javac PI.java
usiellarb@Tashido ~/E/S/Tarea I PI>
```

Figura I: Compilación.

En la *Figura II* se muestra la ejecución de los hilos cliente que se encargarán de realizar los cálculos con los nodos 1, 2 y 3 correspondientemente, debido a que aún no ejecutamos el hilo servidor, estos se mantienen a la espera, comprobando la correcta programación.



```
fish /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI
java /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI 55x11
usiellarb@Tashido ~/E/S/Tarea I PI> java PI 1

fish /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI 57x11
java /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI 57x11
usiellarb@Tashido ~/E/S/Tarea I PI> java PI 2

fish /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI 57x12
java /home/usiellarb/Escritorio/Sistemas Distribuidos/Tarea I PI 57x12
usiellarb@Tashido ~/E/S/Tarea I PI> java PI 3
```

Figura II: Ejecución y espera de los nodos 1, 2 y 3.



En la *Figura III* finaliza la ejecución de todos los hilos debido a que ya se ejecutó el servidor permitiendo a los clientes realizar los cálculos y acceder a la sección crítica. Se imprimen los datos para llevar un seguimiento de los mismos en cada uno de los programas y al final en el nodo 0 se muestra el resultado obtenido de PI correctamente.

```
fish /home/usielarb/Escritorio/Sistemas Distribuidos/Tarea I PI 50x8
usi el arb@Tashi do ~/E/S/Tarea I PI > java PI 1
9. 436047343801526 1
- 9. 436047343801526 1
usi el arb@Tashi do ~/E/S/Tarea I PI >

fish /home/usielarb/Escritorio/Sistemas Distribuidos/Tarea I PI 53x8
usi el arb@Tashi do ~/E/S/Tarea I PI > java PI 2
8. 785402214017193 2
8. 785402214017193 2
usi el arb@Tashi do ~/E/S/Tarea I PI >

fish /home/usielarb/Escritorio/Sistemas Distribuidos/Tarea I PI 50x8
usi el arb@Tashi do ~/E/S/Tarea I PI > java PI 3
8. 461056380003658 3
- 8. 461056380003658 3
usi el arb@Tashi do ~/E/S/Tarea I PI >

fish /home/usielarb/Escritorio/Sistemas Distribuidos/Tarea I PI 53x8
usi el arb@Tashi do ~/E/S/Tarea I PI > java PI 0
12. 253294138380149 8. 785402214017193
Val or de PI : 3. 141592628592157
usi el arb@Tashi do ~/E/S/Tarea I PI >
```

Figura III. Ejecución del nodo 0 obteniendo el resultado.

Código:

```
import java.net.Socket;
import java.net.ServerSocket;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.lang.Thread;
import java.nio.ByteBuffer;

import jdk.dynalink.beans.StaticClass;

class PI
{
    static Object lock = new Object();
    static double pi = 0;
    static class Worker extends Thread{
        Socket connexion;
        Worker(Socket connexion)
        {
            this.connexion = connexion;
        }
        public void run()
```



```
{
    // Algoritmo 1
    try{
        DataOutputStream salida = new
DataOutputStream(conexion.getOutputStream());
        DataInputStream entrada = new
DataInputStream(conexion.getInputStream());
        double x;
        x = entrada.readDouble();
        synchronized(lock)
        {
            pi = x + pi;
        }
        salida.close();
        entrada.close();
        conexion.close();
    }
    catch (Exception e)
    {
        System.out.println("Error, catch 1");
    }
}
}

public static void main(String[] args) throws Exception
{
    if (args.length != 1)
    {
        System.err.println("Uso:");
        System.err.println("java PI <nodo>");
        System.exit(0);
    }
    int nodo = Integer.valueOf(args[0]);
    if (nodo == 0)
    {
        // Algoritmo 2
        ServerSocket servidor = new ServerSocket(50000);
        Worker [] w = new Worker[3];
        int i = 0;
        for(;;)
        {
            if (i==3)
                break;
        }
    }
}
```



```
        Socket conexion;  
        conexion = servidor.accept();  
        w[i] = new Worker(conexion);  
        w[i].start();  
        i++;  
    }  
    double suma = 0;  
    i = 0;  
  
    for (;;)   
    {  
        if (i == 100000000)  
            break;  
        suma = 4.0/(8*i+1)+ suma;  
        i++;  
    }  
  
    synchronized(lock) {  
        System.out. println(suma + " " + pi);  
        pi = suma + pi;  
    }  
    i=0;  
    for(;;)  
    {  
        if (i==3)  
            break;  
        w[i].join();  
        i++;  
    }  
    System.out.println("Valor de PI: " + pi);  
}  
else  
{  
    // Algoritmo 3  
    Socket conexion = null;  
    for(;;)  
    try  
    {  
        conexion = new Socket("localhost",50000);  
        break;  
    }  
    catch (Exception e)
```



```
{
    Thread.sleep(100);
}
DataOutputStream salida = new
DataOutputStream(conexion.getOutputStream());
DataInputStream entrada = new
DataInputStream(conexion.getInputStream());
double suma = 0;
int i = 0;
for(;;)
{
    if (i == 10000000)
        break;
    suma = 4.0 / (8*i + 2*(nodo-1) + 3) + suma;
    i++;
}
System.out.println(suma + " " + nodo);
suma = nodo%2==0?suma:-suma;
System.out.println(suma + " " + nodo);
salida.writeDouble(suma);
salida.close();
entrada.close();
conexion.close();
}
}
```