////////Day Four

Driver Class

**package** com.niit.PipesExample;

**import** java.io.\*;

**public** **class** Driver

{

**public** **static** **void** main(String args[])

{

**try**

{

/\* set up pipes \*/

PipedOutputStream pout1 = **new** PipedOutputStream();

PipedInputStream pin1 = **new** PipedInputStream(pout1);

PipedOutputStream pout2 = **new** PipedOutputStream();

PipedInputStream pin2 = **new** PipedInputStream(pout2);

/\* construct threads \*/

FServer producer = **new** FServer(pout1);

PPiper Ppiper = **new** PPiper(pin1, pout2);

Client consumer = **new** Client(pin2);

/\* start threads \*/

producer.start();

Ppiper.start();

consumer.start();

// consumer.setDaemon(true);

**try**

{

System.***out***.println("Demon Called");

}

**catch**(Exception h)

{

}

//producer.setDaemon(true);

}

**catch** (IOException e){}

}

}

////////////////////////

FServer

//////////////////

package com.niit.PipesExample;

import java.io.\*;

import java.util.Random;

class FServer extends Thread

{

/\*\*

Constructs a producer thread.

@param os the output stream

\*/

private DataOutputStream out;

private Random rand = new Random();

public FServer(OutputStream os)

{

out = new DataOutputStream(os);

}

public void run()

{

while (true)

{

try

{

double num = rand.nextDouble();

out.writeDouble(num);

out.flush();

sleep(Math.abs(rand.nextInt() % 1000));

}

catch(Exception e)

{

System.out.println("Error: " + e);

}

}

}

}

/////////////////////////////////////

Class PPIpes

**package** com.niit.PipesExample;

**import** java.io.\*;

/\*\*

A thread that reads numbers from a stream and writes their

average to an output stream.

\*/

**class** PPiper **extends** Thread

{

/\*\*

Constructs a filter thread.

**@param** is the output stream

**@param** os the output stream

\*/

**private** DataInputStream in;

**private** DataOutputStream out;

**private** **double** total = 0;

**private** **int** count = 0;

**public** PPiper(InputStream is, OutputStream os)

{

in = **new** DataInputStream(is);

out = **new** DataOutputStream(os);

}

**static** **int** *counter*=0;

**public** **void** run()

{

**for** (;;)

{

**try**

{

**double** x = in.readDouble();

total += x;

count++;

*counter*++;

**if** (count != 0) out.writeDouble(total / count);

}

**catch**(IOException e)

{

System.***out***.println("Error: " + e);

}

}

}

}

////////////////////////

Client.java

**package** com.niit.PipesExample;

**import** java.io.\*;

**class** Client **extends** Thread

{

/\*\*

Constructs a consumer thread.

**@param** is the input stream

\*/

**public** Client(InputStream is)

{

in = **new** DataInputStream(is);

}

**public** **void** run()

{

**for**(;;)

{

**try**

{

**double** x = in.readDouble();

**if** (Math.*abs*(x - oldx) > ***THRESHOLD***)

{

System.***out***.println("Client"+x);

oldx = x;

}

}

**catch**(IOException e)

{

System.***out***.println("Error: " + e);

}

}

}

**private** **double** oldx = 0;

**private** DataInputStream in;

**private** **static** **final** **double** ***THRESHOLD*** = 0.01;

}

/////////////////

DaemonThread

/////////

**package** com.niit.PipesExample;

**public** **class** DaemonThread **extends** Thread {

**public** **void** run()

{

**if**(Thread.*currentThread*().isDaemon())

{

System.***out***.println("This is Daemon thread");

}

**else**

{

System.***out***.println("This is User thread");

}

}

}