TP 3

Partie 1 : Exécuter le programme suivant sur une seule machine, que remarquez-vous :

```
import java.io.*;
import java.net.SocketException;
public class Process extends Thread{
   public void run(){
      try{
         System.out.println("Je suis dans le " + Thread.currentThread().getName());
         Thread.sleep(3000);
         System.out.println ("Le " + Thread.currentThread().getName() + " est términer!");
      }
      catch (InterruptedException e) { }
   }
   public static void main(String args[]){
      System.out.println ("Créer le thread 0");
      Thread T1 = new Process();
      System.out.println ("Créer le thread 1");
      Thread T2 = new Process();
      T1.start();
      T2.start();
   }
}
```

Partie 2 : Connecter maintenant deux machines, exécuter le programme UDPServer sur une seule machine et le programme UDPClient sur les deux machines en même temps, que remarquez-vous :

```
import java.net.*;
import java.io.*;
import java.util.Scanner;
public class UDPClient {
   public static void main (String args[]){
      try {
         while(true){
            System.out.println("Donner l'opérateur : +  -  *  /");
            Scanner clavier = new Scanner(System.in);
            String op = clavier.nextLine();
            System.out.println("Donner le premier nombre");
            int N1 = clavier.nextInt();
            System.out.println("Donner le deuxième nombre");
            int N2 = clavier.nextInt();
            String operation =  N1+":"+op+":"+N2;
            DatagramSocket ds = new DatagramSocket(7000);
            InetAddress AdrClient = InetAddress.getByName("127.0.0.1");
            byte[] p = operation.getBytes();
            DatagramPacket dp = new DatagramPacket(p, p.length, AdrClient, 8000);
            ds.send(dp);
            Thread.sleep(1000);
            byte[]tampon = new byte[1000];
            DatagramPacket dp1 = new DatagramPacket(tampon, tampon.length);
            ds.receive(dp1);
            String s = new String(dp1.getData());
            System.out.println(s);
            ds.close();
         }
      }
      catch (SocketTimeoutException e) { }
      catch(SocketException e) { }
      catch(IOException e) { }
      catch (InterruptedException eeee) { }
   }
}
```

```
import java.net.*;
import java.io.*;
public class UDPServer {
   public static void main (String args[]){
      try{
         while(true){
            DatagramSocket ds = new DatagramSocket(8000);
            byte[] tampon = new byte[1000];
            DatagramPacket dp = new DatagramPacket(tampon, tampon.length);
            ds.receive(dp);
            new ClientThread(ds,dp);
         }
      }
      catch(SocketException e){ }
      catch(IOException e){ }
   }
}
class ClientThread extends Thread{
   DatagramSocket ds;
   DatagramPacket dp;
   public ClientThread(DatagramSocket ds, DatagramPacket dp){
      this.ds=ds;
      this.dp=dp;
      this.start();
   }
   public void run() {
      try{
         String input = new String(dp.getData());
         String[] in = input.split("[:]");
         double temp=0;
         if (in[1].matches("[+]")) temp = Double.valueOf(in[0])+Double.valueOf(in[2]);
         if (in[1].matches("[-]")) temp = Double.valueOf(in[0])-Double.valueOf(in[2]);
         if (in[1].matches("[*]")) temp = Double.valueOf(in[0])*Double.valueOf(in[2]);
         if (in[1].matches("[/]")) temp = Double.valueOf(in[0])/Double.valueOf(in[2]);
        String r ="Résultat = "+temp+" calculer par: "+ Thread.currentThread().getName();
         byte[] m = r.getBytes();
         InetAddress AdrServer = InetAddress.getByName("127.0.0.1");
         DatagramPacket dp = new DatagramPacket(m, m.length, AdrServer, 7000);
         ds.send(dp) ;
      }
      catch(SocketException e){ }
      catch(IOException e){ }
   }
}
```

Partie 3 : Modifiez les programmes pour que l'UDPServer calcule le PGCD ou le PPCM de deux nombres entiers au choix de l'UDPClient.