```java
/*
 *  Cette classe se connecte à un serveur qui affiche la sinusoïde d'amplitude e
t de
 *  phase fournies en paramètres programme
 *
 *  Usage : java Client host port amplitude phase
 *  Note : mode connecté
 *
 * @author Vincent Granet (Vincent.Granet@univ-cotedazur.fr)
 */
import java.net.*;
import java.io.*;
import java.util.*;

public class Client {
    public static void main (String [] args) {
        String Usage = "Usage: java Client host port amplitude phase";
        // vérifier le nombre de paramètres programme
        if (args.length != 4) {
            System.err.println(Usage);
            System.exit(1);
        }
        // bon nombre de paramètres programme
        try (Socket socket = new Socket(args[0], Integer.parseInt(args[1]));
             DataOutputStream out =
             new DataOutputStream(socket.getOutputStream()))
            {
                double amplitude = Double.valueOf(args[2]);
                double phase = Double.valueOf(args[3]);
                out.writeDouble(amplitude);
                out.writeDouble(phase);
            }
        catch (Exception e) { e.printStackTrace(); }
    }
}
```



```java
import java.net.*;
import java.io.*;
import java.util.Date;
import java.util.function.*;

/**
 * gère la connexion avec un client particulier.
 */
public class GestionDuClient implements Runnable {
    private Thread proc = new Thread(this);
    private Socket socket;
    Plan2D plan;
    // le constructeur
    public GestionDuClient(Socket s, Plan2D plan) {
        this.socket = s;
        this.plan = plan;
        // démarrer le thread courant
        this.proc.start();
    }

    @Override
    public void run() {
        try (DataInputStream in = new DataInputStream(socket.getInputStream()))
            {
                double amplitude = in.readDouble();
                double phase = in.readDouble();
                DoubleFunction<Double> func = x -> amplitude*Math.sin(x+phase);
                for (double x=plan.getMinAbcisse(); x<=plan.getMaxAbcisse(); x+=
0.01) {
                    try {
                        Thread.sleep(10);
                        // le plan est partagé par différents threads
                        // c'est donc une ressource critique => la protéger.
                        synchronized (plan) {
                            // tracer le point (x,y) sur le plan
                            plan.tracerPoint(x, func.apply(x));
                        }
                    }
                    catch (InterruptedException e) {}
                }
                socket.close();
            }
        catch (IOException e) { e.printStackTrace(); }
    }
}
```

```java
/**
 * Cette classe représente un serveur d'affichage de sinusoïdes dans
 * une planche à dessin. Elle accepte des connexions de clients qui
 * fournissent l'amplitude et la phase de la sinusoïde à afficher.  Le
 * port est passé en paramètre, ainsi que la longueur et la hauteur
 * (en pixels) de la planche à dessins.

 Modèle client/serveur
 *
 * @author Vincent Granet (vg@unice.fr)
 */

import java.net.*;
import java.io.*;
import PaD.PlancheADessin;

public class Serveur {
    public static void main (String [] args) {
        String Usage = "Usage: java Serveur port longueur hauteur";
        // vérifier le nombre de paramètres programme
        if (args.length != 3) {
            System.err.println(Usage);
            System.exit(1);
        }
        // bon nombre de paramètres programme
        int port = Integer.parseInt(args[0]);
        // creation de la socket serveur
        try (ServerSocket socketServeur = new ServerSocket(port)) {
            // créer la planche à dessins décorée par PlanD2
            Plan2D plan = new Plan2D(Integer.valueOf(args[1]), Integer.valueOf(a
rgs[2]));
            while (true) {
                // attendre la prochaine connexion
                Socket socketClient = socketServeur.accept();
                // gérer le client dans un thread
                new GestionDuClient(socketClient, plan);
            }
        }
        catch (IOException e) { e.printStackTrace(); }
    }
}
```