Implementar un servidor Socket concurrente y que atienda hasta 100 llamadas de clientes Sockets para el juego del ahorcado.

Para ello, en el proyecto base se facilita el Cliente, que no es concurrente, sino que lanza un único proceso con el que interactúa con el servidor (este cliente puede ser invocado múltiples veces, ya que el servidor es concurrente y tiene un límite muy alto de clientes simultáneos).

Se deberán implementar las clases MainServidor.java y Servidor.java, de tal forma que:

MainServidor.java cree un nuevo hilo para dar respuesta a todos los clientes que se generen (hasta 100 simultáneamente).

Servidor.java que conteste las preguntas enviadas por un cliente con la evolución del juego.

MainServidor (la que tendría que hacer)

```
package Ejercicio6;
import java.io.IOException;
import java.net.*;
import java.util.ArrayList;


public class MainServidor {


   public static void main(String[] args) throws InterruptedException {
      int puertoServidor=6000;
      ArrayList<Thread> hilosServidor = new ArrayList();
      ServerSocket socketServidor;
      int numCliente = 0;

      try {
         // Punto de conexión del servidor (socket en el puerto indicado en la máquina donde se ejecute el proceso)
         socketServidor = new ServerSocket(puertoServidor);

         // Mensaje de arranque de la aplicación
         System.out.println("SERVIDOR CONCURRENTE");
         System.out.println("--------------------");
         System.out.println("Servidor iniciado.");
         System.out.printf("Escuchando por el puerto %d.\n", socketServidor.getLocalPort());
         System.out.println("Esperando conexión con cliente.");


         while (numCliente < 100) {

            // Quedamos a la espera ("escuchando") de que se realice una conexión con el socket de servidor.
            // En el momento en que eso suceda, se aceptará. Mientras tanto, la ejecución queda aquíe
            // bloqueada en espera a que se reciba esa petición por parte de un cliente.
            Socket clientSocket = socketServidor.accept();

            //Interacción del servidor con un cliente
            System.out.println("Conexión establecida con cliente."); // Debug

            // Creamos un nuevo hilo de ejecución para servir a este nuevo cliente conectado
            Servidor hiloServidor = new Servidor(clientSocket, numCliente + 1);
            hilosServidor.add(hiloServidor);
            numCliente++;

            // Lanzamos la ejecución de ese nuevo hilo
            hiloServidor.start();
         } // y seguimos "escuchando" a otras posibles peticiones de cliente

         //Cuando finalice cierro el socket del servidor
         socketServidor.close();

      } catch (SocketException ex) {
         System.out.printf("Error de socket: %s\n", ex.getMessage());
      } catch (IOException ex) {
         System.out.printf("Error de E/S: %s\n", ex.getMessage());
      }

      System.out.println("Fin de ejecución del servidor.");
   }
}
```

Servidor (la que tendría que hacer)

```
package Ejercicio6;

import java.io.*;
import java.net.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;


public class Servidor extends Thread {

    private Socket socketCliente;

    // Flujo de salida a través del cual enviaremos información al proceso cliente
    // conectado a través del socket
    private DataOutputStream flujoEscrituraCliente;

    // Flujo de entrada a través del cual recibiremos información desde el proceso cliente
    private DataInputStream flujoEntradaCliente;

    private int numCliente;

    private String[]posiblesCadenas = {"Futbol","Mesa","Invitacion","Clasificacion","Esternocleidomastoideo"};

    private String incognita;
    private String progreso;
    private int numFallos;
    private boolean acertado;

    public Servidor(Socket socketCliente, int numCliente) throws IOException {
        this.socketCliente = socketCliente;
        // Flujo de salida a través del cual enviaremos información al proceso cliente
        // conectado a través del socket
        this.flujoEscrituraCliente = new DataOutputStream(this.socketCliente.getOutputStream());

        // Flujo de entrada a través del cual recibiremos información desde el proceso cliente
        this.flujoEntradaCliente = new DataInputStream(this.socketCliente.getInputStream());

        this.numCliente = numCliente;

        Random rd = new Random();
        int n = rd.nextInt(5);
        incognita = posiblesCadenas[n];
        progreso=""+incognita.charAt(0);
        for(int i=1;i<incognita.length();i++){
            progreso=progreso+"-";
        }
        System.out.println("--------->"+incognita);
        System.out.println("--------->"+progreso);
        acertado=false;
        numFallos=0;
    }

    @Override
    public void run() {

        String resultado, peticionCliente = null,respuesta;

        System.out.printf("Iniciado juego en el hilo del servidor %d.\n", this.numCliente);

        try {
            while(numFallos<5 && acertado == false){
                peticionCliente = flujoEntradaCliente.readUTF();
                if (!peticionCliente.isEmpty()) {
                    System.out.println("Recicienbo del CLIENTE: \n\t" + peticionCliente);
                    respuesta=calcularRespuesta(peticionCliente);
                    flujoEscrituraCliente.writeUTF(respuesta);
                }
            }
```

```
            // Cerramos la comunicación con el cliente
            flujoEscrituraCliente.close();
            flujoEntradaCliente.close();
            this.socketCliente.close();

        } catch (SocketException ex) {
            System.out.printf("Error de socket: %s\n", ex.getMessage());
        } catch (IOException ex) {
            System.out.printf("Error de E/S: %s\n", ex.getMessage());
        }

        System.out.printf(
                "Hilo servidor %d: Fin de la conexión con el cliente.\n", this.numCliente);

    }

    private String calcularRespuesta(String peticionCliente) {
        char letra = peticionCliente.charAt(0);
        boolean encontrado=false;
        String respuesta="";
        for(int i=0; i<this.incognita.length();i++){
            if(incognita.charAt(i)==letra){
                progreso = progreso.substring(0,i)+letra+progreso.substring(i+1);
                encontrado = true;
            }
        }
        if(encontrado){
            if(incognita.equals(progreso)==true){
                respuesta = "Has acertado la palabra. Fin del programa.- "+progreso;
                acertado=true;
            }else{
                respuesta = "Letra correcta.- "+progreso;
            }
        }else{
            numFallos++;
            if(numFallos<5){
                respuesta = "Letra incorrecta.- "+progreso;
            }else{
                respuesta = "Game over.-";
            }
        }
        return respuesta;
    }
}
```

MainCliente (me la dan)

```
package Ejercicio6;

import java.io.*;
import java.net.Socket;
import java.util.*;

public class MainCliente extends Thread {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("PROGRAMA CLIENTE INICIADO ...");
        Socket cliente = null;
        //CREO FLUJO DE ENTRADA AL SERVIDOR
        DataInputStream flujoEntrada = null;
        //CREO FLUJO DE SALIDA AL SERVIDOR
        DataOutputStream flujoSalida = null;
        String respuesta = "";
        String letra;

        try {
            cliente = new Socket("localhost", 6000);

            flujoSalida = new DataOutputStream(cliente.getOutputStream());
            flujoEntrada = new DataInputStream(cliente.getInputStream());

            while (respuesta.contains("Game over") == false && respuesta.contains("Has acertado la palabra") == false) {
                //ENVÍO UN SALUDO AL SERVIDOR
                System.out.print("Dame una letra: ");
                letra = sc.nextLine();
                flujoSalida.writeUTF(letra);
                respuesta = flujoEntrada.readUTF();
                //EL SERVIDOR ME ENVÍA UN MENSAJE
                System.out.println("Recibiendo del SERVIDOR: \n\t" + respuesta);
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        //CERRAR SREAMS Y SOCKETS
        try {
            flujoEntrada.close();
            flujoSalida.close();
            cliente.close();
        } catch (IOException io) {
            io.printStackTrace();
        }
    }
}
```