Tarea para PSP 05

Roy Denys Calderon Manrique

Ejercicio 2

Modificar el Servidor HTTP

-Atendiendo a la explicación del tema correspondiente, para que nuestro servidor pueda gestionar las peticiones de los clientes de manera concurrente y eficiente, debemos hacer uso de la clase Thread, de modo que el servidor acepte las solicitudes de los clientes para hacer uso de los recursos del servidor y este (Servidor) cree un hilo que se encargue de realizar las peticiones del cliente por cada uno de clientes que vayan entrando al servidor

5.2.- Implementar comunicaciones simultáneas.

Para proporcinar la funcionalidad a nuestro servidor, de que pueda atender comunicaciones simultáneas es necesario utilizar hilos o threads.

Un servidor HTTP realista tendrá que atender varias peticiones simultáneamente. Para ello, tenemos que ser capaces de modificar su código para que pueda utilizar varios hilos de ejecución. Esto se hace de la siguiente manera:

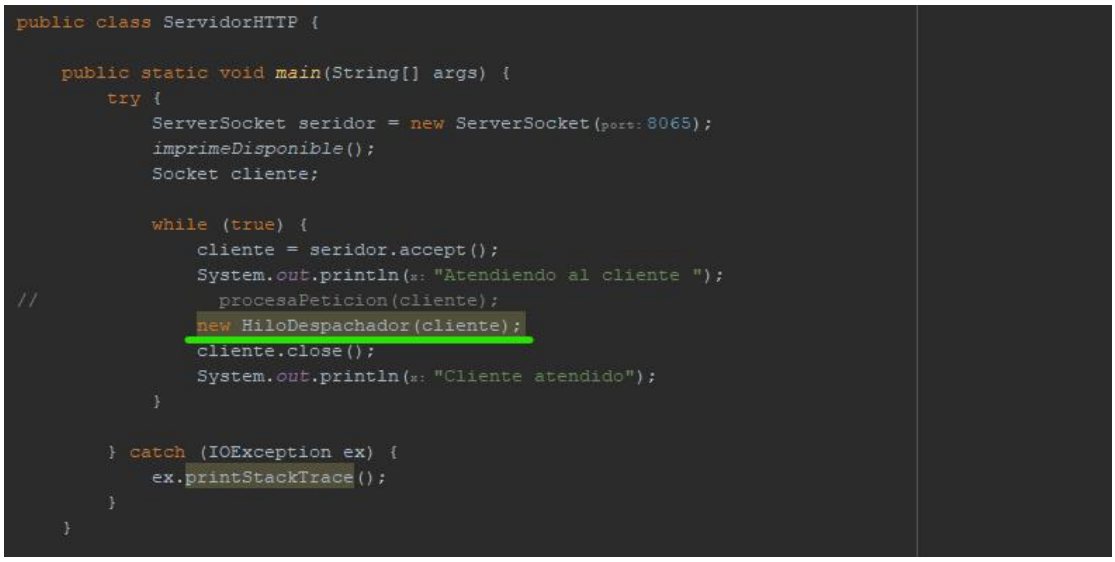
- El hilo principal (o sea, el que inicia la aplicación) creará el `socket` servidor que permanecerá a la espera de que llegue alguna petición.
- Cuando se reciba una, la aceptará y le asignará un `socket` cliente para enviarle la respuesta. Pero en lugar de atenderla él mismo, el hilo principal creará un nuevo hilo para que la despache por el `socket` cliente que le asignó. De esta forma, podrá seguir a la espera de nuevas peticiones.

Esquemáticamente, **el código del hilo principal tendrá el siguiente aspecto**:

```
try {
  socServidor = new ServerSocket(puerto);
   while (true) {
    //acepta una petición, y le asigna un socket cliente para la respuesta
    socketCliente = socServidor.accept();
    //crea un nuevo hilo para despacharla por el socketCliente que le asignó
    hilo = new HiloDespachador(socketCliente);
    hilo.start();
   }
  } catch (IOException ex) {
 }
```

donde la clase `HiloDespachador` será una extensión de la clase `Thread` de Java, cuyo constructor almacenará el `socketCliente` que recibe en una variable local utilizada luego por su método `run()` para tramitar la respuesta:

-Poniendo en practica esto, lo que debemos hacer sera crear un hilo a partir de una clase que implemente la clase Thread, y pasando como parámetro el socket cliente para recibir sus peticiones y realizarlas



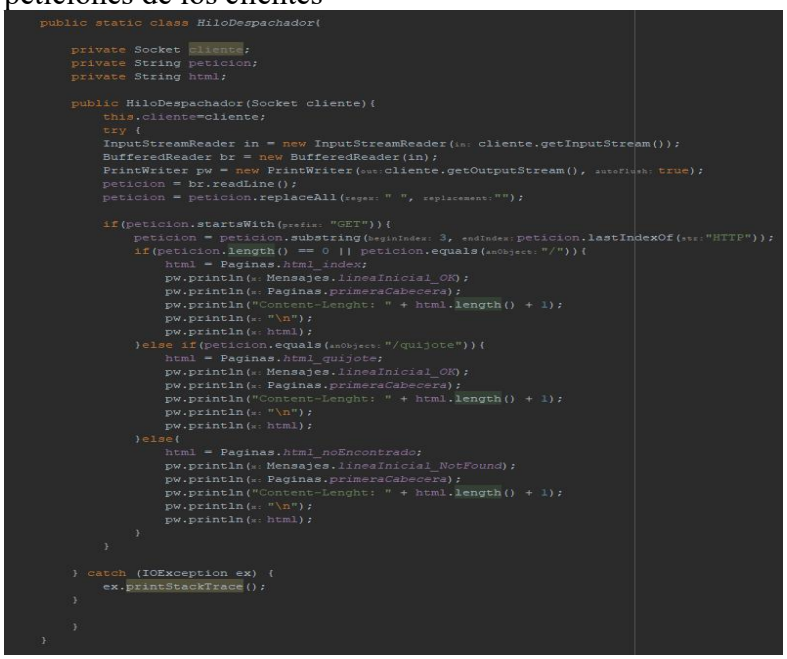
```
public class ServidorHTTP {

    public static void main(String[] args) {
        try {
            ServerSocket seridor = new ServerSocket(port: 8065);
            imprimeDisponible();
            Socket cliente;

            while (true) {
                cliente = seridor.accept();
                System.out.println(x: "Atendiendo al cliente ");
//                  procesaPeticion(cliente);
                new HiloDespachador(cliente);
                cliente.close();
                System.out.println(x: "Cliente atendido");
            }

        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
```

- Y por ultimo debemos configurar la clase HiloDespachador para que gestione las peticiones de los clientes



```
public static class HiloDespachador(

    private Socket cliente;
    private String peticion;
    private String html;

    public HiloDespachador(Socket cliente){
        this.cliente=cliente;
        try {
        InputStreamReader in = new InputStreamReader(in: cliente.getInputStream());
        BufferedReader br = new BufferedReader(in);
        PrintWriter pw = new PrintWriter(out:cliente.getOutputStream(), autoFlush: true);
        peticion = br.readLine();
        peticion = peticion.replaceAll(regex: " ", replacement:"");

        if(peticion.startsWith(prefix: "GET")){
            peticion = peticion.substring(beginIndex: 3, endIndex:peticion.lastIndexOf(str:"HTTP"));
            if(peticion.length() == 0 || peticion.equals(anObject: "/")){
                html = Paginas.html_index;
                pw.println(x: Mensajes.lineaInicial_OK);
                pw.println(x: Paginas.primeraCabecera);
                pw.println("Content-Lenght: " + html.length() + 1);
                pw.println(x: "\n");
                pw.println(x: html);
            }else if(peticion.equals(anObject: "/quijote")){
                html = Paginas.html_quijote;
                pw.println(x: Mensajes.lineaInicial_OK);
                pw.println(x: Paginas.primeraCabecera);
                pw.println("Content-Lenght: " + html.length() + 1);
                pw.println(x: "\n");
                pw.println(x: html);
            }else{
                html = Paginas.html_noEncontrado;
                pw.println(x: Mensajes.lineaInicial_NotFound);
                pw.println(x: Paginas.primeraCabecera);
                pw.println("Content-Lenght: " + html.length() + 1);
                pw.println(x: "\n");
                pw.println(x: html);
            }
        }

    } catch (IOException ex) {
        ex.printStackTrace();
    }

    }
}
```

Ejercicio 1

Modificar Servido HTTP para que incluya la cabecera Date

-Siguiendo las instrucciones del temario, para incorporar la cabecera Date dentro de nuestro servidor HTTP debemos hacer lo siguiente
1. Esta cabecera suele ir justo antes de la linea de separacion (“\n”), para que el registro del inicio de la transmision sea lo mas fiel posible

```
html = Paginas.html_index;
pw.println(x: Mensajes.lineaInicial_OK);
pw.println(x: Paginas.primeraCabecera);
pw.println("Content-Lenght: " + html.length() + 1);
pw.println("Date: " + mostrarHora());
pw.println(x: "\n");
pw.println(x: html);
```

2. El formato de la fecha es inmodificable

```
private static String mostrarHora() {
        SimpleDateFormat sdf = new SimpleDateFormat(pattern:"EEE, dd MMM yyyy HH:mm:ss zzz", locale: Locale.US);
    sdf.setTimeZone(zone:TimeZone.getTimeZone(ID: "GMT"));
    return sdf.format(new Date());
}
```

3. Incluir el metodo mostrarHora() y ejecutar el programa, ahora nuestro programa registrara la hora a la que fue iniciada la transmision