ACTIVIDAD 2 _SOCKETS

GITHUB: https://github.com/Frida-abella/Act2_PSP_Sockets

David Matías Mota

Jaime Aranda Congil

Frida Abella Fernández

Se pide hacer un programa cliente-servidor con sockets, dicho programa consistirá en crear una aplicación que gestione una serie de libros de una biblioteca virtual.

Primero, creamos la clase Libro que tiene: ISBN, AUTOR, TITULO Y PRECIO. Creamos su constructor además de getter y setter para acceder a ella.

```java
package SocketServidor;

public class Libro {

    private String isbn;
    private String autor;
    private String titulo;
    private String precio;

    public Libro(String isbn, String autor, String titulo, String precio) {
        super();
        this.isbn = isbn;
        this.autor = autor;
        this.titulo = titulo;
        this.precio = precio;

    }

    public Libro() {

    }

    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getPrecio() {
        return precio;
    }

    public void setPrecio(String precio) {
        this.precio = precio;
    }

    @Override
    public String toString() {
        return "libro [isbn=" + isbn + ", autor=" + autor + ", titulo=" + titulo + ", precio=" + precio + "]";
    }

}
```

Clase Libro

Después creamos la clase Biblioteca en la que estará la lista predeterminada de libros que guardará el servidor en memoria. Creamos un ArrayList de los diferentes arrays Libro. Además, tiene varios métodos:

1- cargarBiblioteca(): Inicializa y forma la biblioteca base (Se pondrá en marcha en el servidor)
2- buscarTitulo(): Recibe un String con el nombre a buscar y a través de un for recorre la listaLibros (ArrayList) y encuentra el libro con el título a buscar.
3- buscarISBN(): Recibe un String con el ISBN a buscar y a través de un for recorre la listaLibros (ArrayList) y encuentra el libro con el ISBN correspondiente.

4- buscarAutor(): Recibe un String con el autor a buscar. Creamos un ArrayList nuevo en el que vamos a ir añadiendo los diferentes libros del autor correspondiente que va encontrando el for. (REQUERIMIENTO 2)

5- añadirLibro(): Recibe un objeto Libro y lo añade al ArrayList de Libros que creamos antes y después muestra la lista al completo. (REQUERIMIENTO 3)

```java
public class Biblioteca {

    //LISTA DE LIBROS PREDETERMINADA
    //SE GUARDAN EN UN ARRAY LIST PARA CONSULTA

    Libro l1 = new Libro("3454", "ELOY MORENO", "TIERRA", "30");
    Libro l2 = new Libro("3455", "YUVAL NOAH", "SAPIENS", "40");
    Libro l3 = new Libro("3456", "JKR", "HARRY POTTER Y LA PIEDRA FILOSOFAL", "20");
    Libro l4 = new Libro("3457", "JULIO VERNE", "ESCUELA DE ROBINSONES", "20");
    Libro l5 = new Libro("3458", "JKR", "HARRY POTTER Y LA CAMARA SECRETA", "15");
    Libro l6 = new Libro("3459", "JULIO VERNE", "VIAJE AL CENTRO DE LA TIERRA", "30");
    Libro l7 = new Libro("3460", "JULIO VERNE", "LA ISLA MISTERIOSA", "17");

    List<Libro> listaLibros = new ArrayList<Libro>();

    public void cargarBiblioteca() {

        listaLibros.add(l1);
        listaLibros.add(l2);
        listaLibros.add(l3);
        listaLibros.add(l4);
        listaLibros.add(l5);
        listaLibros.add(l6);
        listaLibros.add(l7);

    }

    public Libro buscarTitulo(String titulo) {
        Libro resultado = null;
        for (Libro l : listaLibros) {
            if (l.getTitulo().equals(titulo)) {
                resultado =  l;
            }
        }
        return resultado;
    }


    public Libro buscarISBN(String isbn) {
        Libro resultado = null;
        for (Libro l : listaLibros) {
            if (l.getIsbn().equals(isbn)) {
                resultado = l;

            }

        }
        return resultado;
    }

    public List<Libro> buscarAutor(String autor) {
        List<Libro> resultado = new ArrayList<>();

        for (Libro l : listaLibros) {
            if (l.getAutor().equals(autor)) {
                resultado.add(l);
            }
        }
        return resultado;

    }

    public void añadirLibro (Libro libro) {


        listaLibros.add(libro);

        System.out.println("Nuevo libro añadido: " + libro.getTitulo());

        System.out.println(listaLibros);

    }


}
```

Clase Biblioteca

Continuamos creando la clase Servidor. Esta crea un objeto Biblioteca y acto seguido crea la lista predeterminada con el método cargarBiblioteca(). Con ello la lista siempre estará disponible mientras el servidor esté activo. Creamos la variable petición que llevará un conteo de la cantidad de peticiones que se le hacen al servidor mientras esté activo. Hemos creado todos los objetos ServerSocket e InetSocketAdress con los que se crea la posibilidad de conectar con él. Establecimos el puerto 2021 con punto de conexión. Un bucle while infinito para que solo pueda pararse de forma manual el servidor va a contener el socket al cliente para esperar su accept() y creamos un Objeto Hilos al que le pasamos la información que nos da el cliente y el objeto biblioteca para que pueda hacer uso de él.

```
public class Servidor {


    public static final int PUERTO = 2021;



    public static void main(String[] args) {


        System.out.println("      APLICACIÓN DE SERVIDOR BIBLIOTECA     ");
        System.out.println("--------------------------------------------");

        Biblioteca biblioteca = new Biblioteca();
        biblioteca.cargarBiblioteca();
        int peticion = 0;

        try (ServerSocket servidor = new ServerSocket()){
            InetSocketAddress direccion = new InetSocketAddress(PUERTO);
            servidor.bind(direccion);

            System.out.println("SERVIDOR: Esperando peticion por el puerto " + PUERTO);

            while (true) {
                //CREA UN SOCKET DIFERENTE POR CADA PETICION DE CLIENTE
                Socket socketAlCliente = servidor.accept();
                System.out.println("SERVIDOR: peticion numero " + ++peticion + " recibida");
                /*ABRIMOS UN HILO NUEVO Y LIBERAMOS EL HILO MAIN PARA QUE ATIENDA
                 *  PETICIONES DE OTROS CLIENTES */
                new Hilos(socketAlCliente, biblioteca);
            }
        } catch (IOException e) {
            System.err.println("SERVIDOR: Error de entrada/salida");
            e.printStackTrace();
        } catch (Exception e) {
            System.err.println("SERVIDOR: Error");
            e.printStackTrace();
        }



    }

}
```

Clase Servidor

A continuación, creamos la clase Cliente. Esta contiene la aplicación del menú con el que interactúa el usuario. Establecemos el mismo puerto que con servidor. Creamos el

objetos Socket y realizamos un connect(). Creamos el canal de comunicación entrada/salida con InputStreamReader y PrintStream y creamos las diferentes variables que vamos a utilizar.

```java
public class Cliente {

    public static final int PUERTO = 2021;
    public static final String IP_SERVER = "localhost";

    public static void main(String[] args) {

        System.out.println("    APLICACIÓN CLIENTE BIBLIOTECA     ");
        System.out.println("--------------------------------------");

        InetSocketAddress direccionServidor = new InetSocketAddress(IP_SERVER, PUERTO);

        try (Scanner sc = new Scanner(System.in)){

            System.out.println("CLIENTE: Esperando a que el servidor acepte la conexión");
            Socket socketAlServidor = new Socket();
            socketAlServidor.connect(direccionServidor);
            System.out.println("CLIENTE: Conexion establecida... a " + IP_SERVER +
                    " por el puerto " + PUERTO);

            InputStreamReader entrada = new InputStreamReader(socketAlServidor.getInputStream());
            BufferedReader entradaBuffer = new BufferedReader(entrada);

            PrintStream salida = new PrintStream(socketAlServidor.getOutputStream());


            //Variable que controla las opciones del switch del menu
            String opcion;

            String isbn = "";
            String titulo = "";
            String autor = "";
            String datosLibro = "";
            //Variable que mandamos al servidor a buscar
            String buscar;
```

Inicio Clase Cliente

Comenzamos el método menú() que contiene el menú de opciones:

```java
private static void menu() {
    System.out.println();
    System.out.println("--------------------------");
    System.out.println("Bienvenido a la Biblioteca");
    System.out.println("--------------------------");
    System.out.println("Escriba el numero de la opcion que desee o FIN para salir");
    System.out.println("1 = Buscar libro por ISBN");
    System.out.println("2 = Buscar libro por TITULO");
    System.out.println("3 = Buscar libros por AUTOR");
    System.out.println("4 = Añadir libro a la biblioteca");
    System.out.println("FIN = Salir del programa");
    System.out.println("--------------------------");
}
```

La variable opción va a recibir por consola la opción a escoger. Que después recoger un switch con el que establecemos las diferentes opciones del menú.

```
do {
    //LLAMAMOS AL METODO MENU PARA DESPLEGAR OPCIONES
    menu();
    //frase que vamos a mandar para contar
    opcion = sc.nextLine().toUpperCase();
    switch (opcion) {
        case "1":
            System.out.println("Escribe el IBSN del libro: ");
            isbn = sc.nextLine();
            opcion = "1";

            buscar  = opcion + "-" + isbn;

            salida.println(buscar);
            System.out.println("CLIENTE: Esperando respuesta ..... ");

            String respuesta1 = entradaBuffer.readLine();
            System.out.println("CLIENTE: Servidor responde, " + respuesta1);

            break;

        case "2":
            System.out.println("Escribe el título del libro: ");
            titulo = sc.nextLine();
            opcion = "2";

            buscar = opcion + "-" + titulo;

            salida.println(buscar);
            System.out.println("CLIENTE: Esperando respuesta ..... ");

            String respuesta2 = entradaBuffer.readLine();
            System.out.println("CLIENTE: Servidor responde, " + respuesta2);

            break;
```

Dentro de cada opción se recoge por teclado la petición (el título, el ISBN o el autor a buscar). Hemos decidido recoger el número de la opción escogida y unirlo al String a buscar con un “-“entre medias. Con esto queríamos enviarle en un solo String al Servidor (Hilos) tanto la opción escogida (para poder hacer diferenciación dentro de Hilos) como la información en sí que luego separamos con un método Split(“-“).

El caso de Añadir un nuevo Libro es diferente puesto que se ha de escribir la información del libro a añadir seguido y separada por “-“. Todo ello recogido en un while del que se sale si el usuario escribe FIN.

```
            case "2":
                System.out.println("Escribe el título del libro: ");
                titulo = sc.nextLine();
                opcion = "2";

                buscar = opcion + "-" + titulo;

                salida.println(buscar);
                System.out.println("CLIENTE: Esperando respuesta ..... ");

                String respuesta2 = entradaBuffer.readLine();
                System.out.println("CLIENTE: Servidor responde, " + respuesta2);

                break;

            case "3":
                System.out.println("Escribe el nombre del autor: ");
                autor = sc.nextLine();
                opcion = "3";

                buscar = opcion + "-" + autor;

                salida.println(buscar);
                System.out.println("CLIENTE: Esperando respuesta ..... ");

                String respuesta3 = entradaBuffer.readLine();
                System.out.println("CLIENTE: Servidor responde, " + respuesta3);

                break;

            case "4":
                System.out.println("Escribe los datos de la siguiente forma"
                        + " ----> ISBN-AUTOR-TITULO-PRECIO <----  "
                        + "del libro que quieres añadir a la biblioteca separado por - ");
                datosLibro = sc.nextLine();
                opcion = "4";

                buscar = opcion + "-" + datosLibro;

                salida.println(buscar);
                System.out.println("CLIENTE: Esperando respuesta ..... ");

                String respuesta4 = entradaBuffer.readLine();
                System.out.println("CLIENTE: Servidor responde, " + respuesta4);

                break;


            case "FIN":
                System.out.println("Programa finalizado");


                salida.println(opcion);
                System.out.println("CLIENTE: Esperando respuesta ..... ");

                String despedida = entradaBuffer.readLine();
                System.out.println(despedida + " ADIOS");

                break;

            default:
                System.out.println("Opción incorrecta");

        }


    } while (!opcion.equals("FIN"));
    //CERRAMOS EL SOCKET
    socketAlServidor.close();

} catch (UnknownHostException e) {
    System.err.println("CLIENTE: No encuentro el servidor en la dirección" + IP_SERVER);
    e.printStackTrace();
} catch (IOException e) {
```

Resto de la Clase Cliente

Por último, creamos la Clase Hilos que implementa Runnable para crear los diferentes hilos. Esta recibe la información del usuario por Socket y la biblioteca predeterminada de servidor para hacer búsquedas o realizar cambios.

```
public class Hilos implements Runnable{
    private Thread hilo;
    private static int numCliente = 0;
    private Socket socketAlCliente;
    private Biblioteca biblioteca;

    public Hilos(Socket socketAlCliente, Biblioteca biblioteca) {
        numCliente++;
        hilo = new Thread(this, "Cliente_"+numCliente);
        this.socketAlCliente = socketAlCliente;
        this.biblioteca = biblioteca;
        hilo.start();
    }
```

Clase Hilos

Creamos el constructor que recibe estos parámetros y se arranca con start().

Después hacemos @Override a su método run(). En el que creamos toda la lógica del servidor. Cada vez que un cliente se conecta al servidor se crea un hilo exclusivamente para él y el hilo Main queda a la espera de nuevos clientes. Creamos el canal entrada/salida y después un bucle while. Como comentamos anteriormente Hilos recibe la opción escogida + "-" + información del usuario. Creamos un array que va a contener las diferentes partes, opción en la posición 0 y la información en la posición 1. Y un switch controla cada opción como en el menú. Gracias a que le hemos pasado la información de biblioteca podemos usar los métodos de Biblioteca y buscar la información requerida. En el caso de añadir libro el array es más largo al tener más información (ISBN+autor+titulo+precio) que introdujo el usuario. La posición 0 ya dijimos que es de la opción escogida, así que creamos variables locales para contener las siguientes posiciones del array que corresponden a la diferente información y se la añadimos a un nuevo objeto Libro que después añadimos a la Biblioteca. (Captura en siguiente página)

```java
@Override
public void run() {
    System.out.println("Estableciendo comunicacion con " + hilo.getName());
    PrintStream salida = null;
    InputStreamReader entrada = null;
    BufferedReader entradaBuffer = null;
    String [] datosLibro = null;
    String opcionEscogida = null;

    try {
        //Salida del servidor al cliente
        salida = new PrintStream(socketAlCliente.getOutputStream());
        //Entrada del servidor al cliente
        entrada = new InputStreamReader(socketAlCliente.getInputStream());
        entradaBuffer = new BufferedReader(entrada);

        String texto = null;
        boolean continuar = true;



        //Procesaremos entradas hasta que el texto del cliente sea FIN
        while (continuar) {
            texto = entradaBuffer.readLine();

            datosLibro = texto.split("-");
            opcionEscogida = datosLibro[0];


                String isbnLibro;
                String autorLibro;
                String tituloLibro;
                String precioLibro;


                switch (opcionEscogida) {
                case "1":

                    isbnLibro = datosLibro[1];
                    salida.println(biblioteca.buscarISBN(isbnLibro));
                break;
                case "2":
                    tituloLibro = datosLibro [1];
                    salida.println(biblioteca.buscarTitulo(tituloLibro));
                break;
                case "3":
                    autorLibro = datosLibro [1];
                    salida.println(biblioteca.buscarAutor(autorLibro));
                break;
                case "4":
                    isbnLibro = datosLibro[1];
                    autorLibro = datosLibro[2];
                    tituloLibro = datosLibro[3];
                    precioLibro = datosLibro[4];

                    // Creamos un nuevo objeto libro con los datos que hemos extraído del String recibido del cliente
                    Libro nuevoLibro = new Libro (isbnLibro, tituloLibro, autorLibro, precioLibro);

                    // Damos de alta el libro en la biblioteca con el método altaLibro(), pasándole como argumento
                    // el nuevo libro que hemos creado con los datos que nos ha pasado el cliente
                    biblioteca.añadirLibro(nuevoLibro);

                    salida.println("El libro que se ha añadido es el siguiente:" + nuevoLibro);
                    break;

                case "FIN":
                    salida.println("Fin. Gracias por establecer conexión");
                    continuar = false;

                    break;

                }

         }
```

Método run()