Dosier de Actividades del Tema 3: *Programación de comunicaciones en red*

Desarrollo de Aplicaciones Multiplataforma

Programación de Servicios y Procesos



Ejercicio 1:

Implementar un programa cliente servidor en Java para almacenar las tareas pendientes de un determinado usuario. La funcionalidad del programa será el siguiente:

El servidor realizará las siguientes funciones:

- Inicio del servicio.
- Creación de una estructura arrayList o similar para almacenar las tareas.
- Procesar todas las instrucciones que vienen del cliente y devolver el resultado.
- El servidor no visualizará ningún tipo de información en su terminal.

El cliente realizará las siguientes funciones:

- Conexión al servidor.
- Enviar órdenes y visualitzar el resultado hasta que introduzcamos la palabra end.
- Si el servidor no reconoce la orden mostrará un mensaje de error.

El formato de la instrucción será: orden – parametro.

Órdenes que se pueden enviar al servidor:

- add tarea (añade una tarea a la lista de tareas pendientes)
- remove tarea (borra la tarea de la lista de tareas pendientes)
- list (lista las tareas pendientes)
- count (devuelve el número de tareas)
- end (finaliza la ejecución del programa)

Ejemplo de funcionamiento del servidor.

G:\Mi unidad\java\act2324>java ServidorTareas
... Servidor de Tareas Online ...



Ejemplo de funcionamiento del cliente.

```
G:\Mi unidad\java\act2324>java ClienteTareas
Conectado al Control de Tareas!
Que operación de deseas realizar: (add,count,list,del,end): add - reunion direccion
reunion direccion añadida a la lista de tareas..
Que operación de deseas realizar: (add,count,list,del,end):add - repaso ejercicio 1
repaso ejercicio 1 añadida a la lista de tareas..
Que operación de deseas realizar: (add,count,list,del,end):add - estudio examenes
estudio examenes añadida a la lista de tareas...
Que operación de deseas realizar: (add,count,list,del,end):list
Tareas pendientes:
reunion direccion
repaso ejercicio 1
estudio examenes
Que operación de deseas realizar: (add,count,list,del,end):count
Tienes 3 tareas pendientes
Que operación de deseas realizar: (add,count,list,del,end):del - estudio examenes
estudio examenes borrada de la lista de tareas..
Que operación de deseas realizar: (add,count,list,del,end):list
Tareas pendientes:
reunion direccion
repaso ejercicio 1
Que operación de deseas realizar: (add,count,list,del,end):
```



Código Servidor

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
public class NewServer {
  public static void main(String[] args) {
    try {
      ServerSocket serverSocket();
      InetSocketAddress addr = new InetSocketAddress("localhost", 5060);
      serverSocket.bind(addr);
      while (true) {
        new Thread(new NewServerThread(serverSocket.accept())).start();
      }
    } catch (Exception e) {
      throw new RuntimeException(e);
  }
class NewServerThread implements Runnable {
  private final Socket clientSocket;
  private final PrintStream clientWriter;
  NewServerThread(Socket clientSocket) {
    this.clientSocket = clientSocket;
    this.clientWriter = getPrintStream();
  }
  private PrintStream getPrintStream() {
      return new PrintStream(clientSocket.getOutputStream(), true);
    } catch (Exception e) {
      throw new RuntimeException(e);
```



```
private static final String MENU UserAction =
           "¿Qué operación deseas realizar?" +
           "<LINE BREAK> add - tarea (Añade una tarea a la lista de tareas pendientes)" +
           "<LINE BREAK> del - tarea (Borra la tarea de la lista de tareas pendientes)" +
           "<LINE BREAK> list
                                     (Lista las tareas pendientes)" +
           "<LINE BREAK> count
                                       (Devuelve el número de tareas)" +
           "<LINE BREAK> end
                                      (Finaliza la ejecución del programa)";
  public void run() {
    NewActivity act = new NewActivity();
    try {
      BufferedReader clientReader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream(), StandardCharsets.UTF 8));
      clientWriter.println("¡Estás conectado al Control de Tareas!");
      String[] action = new String[2];
      do {
         clientWriter.println(MENU_UserAction);
         action[0] = clientReader.readLine();
         if (action[0].contains("add - ") | | action[0].contains("del - ")) {
           String[] response = action[0].split(" - ", 2);
           action[0] = response[0];
           action[1] = response[1];
         }
         switch (action[0]) {
           case "add":
             clientWriter.println(act.addActivityToTodoList(action[1]));
             break:
           case "del":
             clientWriter.println(act.delActivityToTodoList(action[1]));
             break;
           case "list":
             clientWriter.println(act.getTodoListActivities());
             break;
           case "count":
             clientWriter.println(act.getTodoListLength());
             break;
           case "end":
             clientWriter.println("Adiós.<LINE BREAK>Buenas tardes.");
```

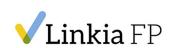


}

```
clientSocket.close();
             break;
           default:
             clientWriter.println("Error.<LINE_BREAK>La acción que has introducido no corresponde
a ninguna acción disponible.<LINE BREAK>");
             break;
         if (clientSocket.isClosed()) break;
      } while (true);
    } catch (Exception e) {
      throw new RuntimeException(e);
  }
class NewActivity {
  private final ArrayList<String> todoList;
  NewActivity() {this.todoList = new ArrayList<>();}
  protected String getTodoListActivities() {
    if (!todoList.isEmpty()) {
      StringBuilder remainingActivities = new StringBuilder("Tareas pendientes:");
      for (int i = 0; i < todoList.size(); i++) {
         remainingActivities.append("<LINE_BREAK>").append(i + 1).append(".
").append(todoList.get(i));
      return remainingActivities.toString();
    }
    return "No hay actividades pendientes.";
  }
  protected String getTodoListLength() {
    return "Tienes" + todoList.size() + " tarea/s pendiente/s.";
  protected String addActivityToTodoList(String activity) {
    todoList.add(activity);
    return "La actividad:<LINE_BREAK>'" + activity + "'<LINE_BREAK>Ha sido añadida a la lista.";
  }
  protected String delActivityToTodoList(String activity) {
    if (todoList.removelf(a -> a.equals(activity))) {
```



```
return "La actividad:<LINE_BREAK>"" + activity + "'<LINE_BREAK>Ha sido eliminada de la lista.";
}
return "No se ha podido encontrar la actividad solicitada.";
}
}
```



Código Cliente

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ConnectException;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
public class NewClient {
  private static final Socket clientSocket = new Socket();
  private static final String hostname = "localhost";
  private static final int port = 5060;
  public static void main(String[] args) {
    InetSocketAddress addr = new InetSocketAddress(hostname, port);
      clientSocket.connect(addr);
      PrintStream serverWriter = new PrintStream(clientSocket.getOutputStream(), true);
      BufferedReader userReader = new BufferedReader(new InputStreamReader(System.in,
StandardCharsets.UTF 8));
      BufferedReader serverReader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream(), StandardCharsets.UTF_8));
      String action;
      printServerResponse(serverReader);
        System.out.println("-----");
        printServerResponse(serverReader);
        System.out.println("Tu acción: ");
        action = userReader.readLine();
        serverWriter.println(action);
        System.out.println();
        printServerResponse(serverReader);
        System.out.println();
      } while (!action.equals("end"));
    } catch (ConnectException ce) {
      System.err.println("Connection refused. No server found listening at " + hostname + ":" +
port);
    } catch (IOException e) {
      throw new RuntimeException(e);
```



```
}

private static void printServerResponse(BufferedReader serverReader) throws IOException {
    String receivedMessage = serverReader.readLine();
    System.out.print("→ ");
    if (receivedMessage.contains("<LINE_BREAK>"))
        for (String received : receivedMessage.split("<LINE_BREAK>")) {
            System.out.println(received);
            }
            else System.out.println(receivedMessage);
        }
}
```



Capturas funcionamiento

(No hago capturas del servidor porque no muestra nada por consola)

Conexión de un cliente al servidor

```
→ ¡Estás conectado al Control de Tareas!

------

→ ¿Qué operación deseas realizar?

add - tarea (Añade una tarea a la lista de tareas pendientes)

del - tarea (Borra la tarea de la lista de tareas pendientes)

list (Lista las tareas pendientes)

count (Devuelve el número de tareas)

end (Finaliza la ejecución del programa)

Tu acción:
```

Añadir tareas a la lista

Listar tareas

```
→ ¿Qué operación deseas realizar?
add - tarea (Añade una tarea a la lista de tareas pendientes)
del - tarea (Borra la tarea de la lista de tareas pendientes)
list (Lista las tareas pendientes)
count (Devuelve el número de tareas)
end (Finaliza la ejecución del programa)
Tu acción:
list

→ Tareas pendientes:
1. preparar la cena
```



Eliminar tareas de la lista

```
→ ¿Qué operación deseas realizar?
add - tarea (Añade una tarea a la lista de tareas pendientes)
del - tarea (Borra la tarea de la lista de tareas pendientes)
list (Lista las tareas pendientes)
count (Devuelve el número de tareas)
end (Finaliza la ejecución del programa)
Tu acción:
del - preparar la cena
→ La actividad:
'preparar la cena'
Ha sido eliminada de la lista.
```

No eliminar tarea

(porque ya ha sido eliminada y no la encuentra)

```
→ ¿Qué operación deseas realizar?
add - tarea (Añade una tarea a la lista de tareas pendientes)
del - tarea (Borra la tarea de la lista de tareas pendientes)
list (Lista las tareas pendientes)
count (Devuelve el número de tareas)
end (Finaliza la ejecución del programa)
Tu acción:
del - preparar la cena
→ No se ha podido encontrar la actividad solicitada.
```

No listar tareas

(Porque no hay tareas guardadas en la lista)

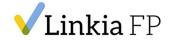
```
→ ¿Qué operación deseas realizar?
add - tarea (Añade una tarea a la lista de tareas pendientes)
del - tarea (Borra la tarea de la lista de tareas pendientes)
list (Lista las tareas pendientes)
count (Devuelve el número de tareas)
end (Finaliza la ejecución del programa)
Tu acción:
list

→ No hay actividades pendientes.
```



Contar tareas

```
→ ¿Qué operación deseas realizar?
 add - tarea
                 (Añade una tarea a la lista de tareas pendientes)
 del - tarea
                 (Borra la tarea de la lista de tareas pendientes)
 list
                 (Lista las tareas pendientes)
                 (Devuelve el número de tareas)
                 (Finaliza la ejecución del programa)
Tu acción:
Tu acción:
→ La actividad:
'preparar la cena'
Ha sido añadida a la lista.
→ ¿Qué operación deseas realizar?
add - tarea (Añade una tarea a la lista de tareas pendientes)
del - tarea
                (Borra la tarea de la lista de tareas pendientes)
                (Lista las tareas pendientes)
list
                (Devuelve el número de tareas)
                (Finaliza la ejecución del programa)
→ Tienes 1 tarea/s pendiente/s.
```



Responder a varios clientes

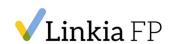
```
C:\repos\DAM_LinkiaFP\M09_Programacion_de_servicios_y_procesos\UF3_Sockets_y_servicios\M09_UF3_T3_Ejercicio1_Client\srcjava java NewClient.java
? ¡Estás conectado al Control de Tareas!

? ¿Qué operación deseas realizar?
add - tarea (Añade una tarea a la lista de tareas pendientes)
del - tarea (Borra la tarea de la lista de tareas pendientes)
list (Lista las tareas pendientes)
count (Devuelve el número de tareas)
end (Finaliza la ejecución del programa)
Tu acción:
```

Tener listas independientes para cada cliente

```
C:\repos\DAM_LinkiaFP\M09_Programacion_de_servicios_y_procesos\UF3_Sockets_y_servici
os\M09 UF3 T3 Ejercicio1 Client\src>java NewClient.java
 ¡Estás conectado al Control de Tareas!
 ¿Qué operación deseas realizar?
                 (Añade una tarea a la lista de tareas pendientes)
 add - tarea
del - tarea
                 (Borra la tarea de la lista de tareas pendientes)
list
                 (Lista las tareas pendientes)
count
                 (Devuelve el número de tareas)
                 (Finaliza la ejecución del programa)
end
Tu acción:
add - comprar pan
 La actividad:
'comprar pan'
Ha sido añadida a la lista.
 ¿Qué operación deseas realizar?
 add - tarea (Añade una tarea a la lista de tareas pendientes)
 del - tarea
                 (Borra la tarea de la lista de tareas pendientes)
                 (Lista las tareas pendientes)
count
                 (Devuelve el número de tareas)
                 (Finaliza la ejecución del programa)
Tu acción:
list
 Tareas pendientes:
 . comprar pan
```

```
\repos\DAM_LinkiaFP\M09_Programacion_de_servicios_y_procesos\UF3_Sockets_y_servici
os\M09 UF3 T3 Ejercicio1 Client\src>java NewClient.java
 ¡Estás conectado al Control de Tareas!
 ¿Qué operación deseas realizar?
                (Añade una tarea a la lista de tareas pendientes)
add - tarea
                (Borra la tarea de la lista de tareas pendientes)
del - tarea
list
                (Lista las tareas pendientes)
count
                (Devuelve el número de tareas)
                (Finaliza la ejecución del programa)
end
Tu acción:
add - comprar lechuga
 La actividad:
'comprar lechuga'
Ha sido añadida a la lista.
 ¿Qué operación deseas realizar?
add - tarea
                (Añade una tarea a la lista de tareas pendientes)
 del - tarea
                 (Borra la tarea de la lista de tareas pendientes)
list
                (Lista las tareas pendientes)
count
                 (Devuelve el número de tareas)
                (Finaliza la ejecución del programa)
 u acción:
 Tareas pendientes:
```



. comprar lechuga

Formato de entrega.

Un archivo comprimido (en formato zip o rar) con el siguiente contenido:

- Archivos .java y jar de los diferentes ejercicios.
- Un documento en formato pdf con las capturas del código y una prueba de funcionamiento de cada actividad.