Frameworks

```
(A) SingleThreadTCPServer
            +startLoop(args: String[]): void
            #checkArguments(args: String[]): void
            #displayUsage(): void
            #displaySocketInformation(portNumber: Int): void
            #acceptAndDisplaySocket(serverSocket: ServerSocket): Socket
            #displaySocketData(clientSocket: Socket): void
            #displayAndExit(portNumber: int): void
            -handleClient(clientSocket: Socket): void
            + «abstract» handleMessage(message: String, out: PrintWriter): void
 1 private final void handleClient(Socket clientSocket) {
       PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
3
4
       BufferedReader in = new BufferedReader(
                              new InputStreamReader(clientSocket.getInputStream());
5
6
       String line;
7 .lem.while ((line = in.readLine()) != null) { do. automos associated as i astro-patological
         System.out.println("Received message: " + inputLine
   + " from " + clientSocket.getInetAddress().getHostAddress()
               + ":" + clientSocket.getPort()); a same seed and band as a during and a
10
11
12 golden if (line.isEmpty()) break; to assess adjuste one several cipo so advancement co.
         this.handleMessage(line, out);
14
       System.out.println("Connection closed with " + clientSocket.getInetAddress().getHostAddress()
15
    + ":" + clientSocket.getPort());
16
17
     } catch (IOException e) {
18
       System.err.println("Error: " + e.getMessage());
19
20
    } finally {
21
       try {
22
         clientSocket.close();
24
    }
25 } anistraco moutro de o checita raional et ne assorre) apud rensalla soligini grinoro de la recent de
```

Inciso 1: Explique cuales son los frozen spot y hot spot del codigo presentado del framework.

Frozen spots: Son aquellas partes del frameworks que son invarientes, es decir, no pueden modificarse, en el codigo presentado los frozen sports van de la linea 2 hasta la 12 y despues van de la linea 15 hasta el fin del metodo. Entre las lineas mencionadas se encuentran los mensajes de conexion y cierre de conexion, que forman parte de los frozen spots.

Hot spots: Son aquellas partes del framework que pueden variar, es decir, permiten modificarse, puede ser a traves de subclases o ganchos, por ejemplo en el codigo presentado, el metodo handleMesage es un metodo abstracto, es un hot spot, esta pensado para que las subclases puedan definir y personalizar el comportamiento del metodo.

Inciso 2: Modificar este framework para permitir personalizar los mensajes que se muestran en consola por defecto cuando un cliente se conecta y se desconecta.

Para hacer esto, extraemos la logica de imprimir los mensajes en dos metodos:

Extraemos la logica para mostrar mensaje de conexion.

```
protected void imprimirMensajeConexion(Socket clientSocket)
{
    System.out.println("Received message: " + inputLine
    + " fron " + clientSocket.getInetAddress().getHostAddress()
    + ":" + clientSocket.getPort());
}
```

Entonces en las lineas 8..10 borramos el codigo, e incovamos al metodo imprimirMensajeConexion(clientSocket)

Extraemos la logica para mostrar mensaje de cierre de conexion

```
protected void imprimirMensajeCierreConexion(Socket clientSocket)
{
    System.out.println("Connection closed with" +
    clientSocket.getInetAddress().getHostAddress()
    + ":" clientSocket.getPort());
}
```

Entonces en las lineas 15..16 borramos el codigo, e incovamos al metodo imprimirMensajeCierreConexion(clientSocket)

Es importante que definamos los metodos como protected, asi dejamos de manera explicita que las subclases pueden redefinir estos metodos para extender o modificar su comportamiento

Inciso 3: Explique brevemente cuales son los frozen spot y los hot sport despues de la extension realizada.

Frozen spots: Despues de la extension, las partes del codigo que son invariantes son:

- linea 2 hasta linea 7
- linea 12,
- linea 18 hasta el fin del metodo.

Hot spots: Despues de la extension, las partes del codigo que son variantes, pueden extenderse, son:

- Linea 8 => metodo imprimirMensajeConexion(clientSocket)
- Linea 15 => metodo imprimirMensajeCierreConexion(clientSocket)

Inciso 4: Indique en donde se produce la inversion de control en la extension realizada.

La inversion de control se produce cuando el framework llama a nuestro codigo. Luego de la extension que hicimos, la inversion de control se generaria en:

- Linea 8 (cuando llama al metodo imprimirMensajeConexion)
- Linea 15 (cuando llama al metodo imprimirMensajeCierreConexion)