```java
/**
 * Mainframe.java
 * <p>
 * Clase que contiene la implementación de los métodos para conectarse con el
 * mainframe y ejecutar la aplicación legada.
 * <p>
 * Radu Constantin Robu y Alberto Pérez
 */
package P3.Control;

import java.io.*;

import static java.lang.Thread.sleep;

public class Mainframe implements MainframeAPI {
    private static final String NOK = "NOK";
    private static final String TASKS2JOB = "tasks2.job";
    private static final String MENU_TASKS2 =
            "TASK MANAGEMENT 2.0 BY TURO-SL SOFT";
    private static final int DELAY = 50;

    private Process proceso;
    private PrintWriter outStream = null;
    private BufferedReader inStream = null;

    private static Mainframe singleton = null;

    /**
     * Constructor de la clase.
     *
     * @throws IOException
     */
    private Mainframe() throws IOException {
        proceso = Runtime.getRuntime().exec(TERMINAL_SIN_PANTALLA);
        inStream = new BufferedReader(new InputStreamReader(
                proceso.getInputStream()));
        outStream = new PrintWriter(new OutputStreamWriter(
                proceso.getOutputStream()));
    }

    /**
     * Devuelve la instancia del objeto de la clase.
     *
     * @return
     * @throws IOException
     */
    public static Mainframe getInstance() throws IOException {
        if (singleton == null) {
            singleton = new Mainframe();
        }
        return singleton;
    }

    /**
     * Realiza la conexión con la máquina.
```

```
56       *
57       * @param host
58       * @return
59       * @throws IOException
60       * @throws InterruptedException
61       */
62      private boolean conectarHost(String host) throws IOException {
63          outStream.println(CONNECT + host);
64          outStream.flush();
65          return ejecutarSiguienteComando();
66      }
67
68      /**
69       * Obtiene la respuesta textual del mainframe en la terminal s3270.
70       *
71       * @return
72       */
73      public String obtenerRespuestaMaquina() throws IOException {
74          String resultado = "";
75          String line = "";
76          do {
77              line = inStream.readLine();
78              if (line == null ||
79                      line.matches(PATRON_RESPUESTA_MAINFRAME_ERROR)) {
80                  return NOK;
81              }
82              resultado += line + "\n";
83          } while (inStream.ready());
84          return resultado;
85      }
86
87      /**
88       * Verifica si se inicia sesión correctamente.
89       *
90       * @return
91       * @throws InterruptedException
92       * @throws IOException
93       */
94      private RESPUESTAS_INICIO_SESION falloInicioSesion()
95              throws InterruptedException, IOException {
96          if (esperarPantalla(MENSAJE_USUARIO_EN_USO)) {
97              if (enviarComando(COMANDO_ENTER)) {
98                  return RESPUESTAS_INICIO_SESION.USUARIO_EN_USO;
99              }
100         } else if (esperarPantalla(MENSAJE_IDIOMA_NO_SOPORTADO)) {
101             if (enviarComando(COMANDO_ENTER)) {
102                 return RESPUESTAS_INICIO_SESION.OK;
103             }
104         } else if (esperarPantalla(MENSAJE_USUARIO_INCORRECTO)) {
105             if (enviarComando(COMANDO_ENTER)) {
106                 return RESPUESTAS_INICIO_SESION.USUARIO_INCORRECTO;
107             }
108         } else if (esperarPantalla(MENSAJE_CONTRASENYA_INCORRECTA)) {
109             if (enviarComando(COMANDO_ENTER)) {
110                 return RESPUESTAS_INICIO_SESION.CONTRASENYA_INCORRECTA;
```

```
111             }
112         }
113         return RESPUESTAS_INICIO_SESION.NOK;
114     }
115 
116     /**
117      * Conexión con el host.
118      *
119      * @param host
120      * @param username
121      * @param password
122      * @return
123      * @throws InterruptedException
124      * @throws IOException
125      */
126     public RESPUESTAS_INICIO_SESION conexion(String host, String username,
127                                              String password)
128             throws InterruptedException, IOException {
129         RESPUESTAS_INICIO_SESION resultadoInicioSesion;
130         if (conectarHost(host)) {
131             if (login(username, password)) {
132                 resultadoInicioSesion = falloInicioSesion();
133                 if (resultadoInicioSesion == RESPUESTAS_INICIO_SESION.OK) {
134                     if (ejecutarTasksJob()) {
135                         return RESPUESTAS_INICIO_SESION.OK;
136                     }
137                 } else {
138                     return resultadoInicioSesion;
139                 }
140             }
141         }
142         return RESPUESTAS_INICIO_SESION.NOK;
143     }
144 
145     /**
146      * Inicio de sesión en el mainframe.
147      *
148      * @param username
149      * @param password
150      * @return
151      * @throws InterruptedException
152      * @throws IOException
153      */
154     private boolean login(String username, String password)
155             throws InterruptedException, IOException {
156         if (esperarPantalla(PANTALLA_CONEXION)) {
157             if (enviarComando(COMANDO_ENTER)) {
158                 if (esperarPantalla(PANTALLA_LOGIN)) {
159                     if (enviarString(username)) {
160                         if (enviarComando(COMANDO_ENTER)) {
161                             if (enviarString(password)) {
162                                 if (enviarComando(COMANDO_ENTER)) {
163                                     return true;
164                                 }
165                             }
```

```
                        }
                    }
                }
            }
        }
        return false;
    }

    /**
     * Ejecuta la aplicación legada.
     *
     * @throws InterruptedException
     * @throws IOException
     */
    public boolean ejecutarTasksJob() throws InterruptedException,
            IOException {
        if (esperarPantalla(PANTALLA_MENU_PRINCIPAL)) {
            if (enviarString(TASKS2JOB)) {
                if (enviarComando(COMANDO_ENTER)) {
                    if (esperarPantalla(MENU_TASKS2)) {
                        return true;
                    }
                }
            }
        }
        return false;
    }

    /**
     * Realiza la salida controlada del mainframe a través de la terminal
     * s3270.
     *
     * @return
     * @throws IOException
     * @throws InterruptedException
     */
    @Override
    public boolean logout() throws IOException {
        enviarString(CADENA_OFF);
        enviarComando(COMANDO_ENTER);
        enviarComando(COMANDO_EXIT);

        return true;
    }

    /**
     * Método sobreescrito para enviar un comando al mainframe.
     *
     * @param comando
     * @return
     * @throws IOException
     */
    @Override
    public boolean enviarComando(String comando) throws IOException {
        outStream.println(comando);
```

```
221         outStream.flush();
222         if (comando.equals(COMANDO_ASCII)) {
223             return true;
224         } else {
225             return ejecutarSiguienteComando();
226         }
227     }
228 
229     /**
230      * Envía una cadena de texto a la terminal s3270.
231      *
232      * @param mensaje
233      * @return
234      * @Override
235      */
236     @Override
237     public boolean enviarString(String mensaje) throws IOException {
238         outStream.println(String.format(FORMATO_CADENA_TEXTO, mensaje));
239         outStream.flush();
240         return ejecutarSiguienteComando();
241     }
242 
243     /**
244      * Método sobreecrito para esperar una pantalla del emulador.
245      *
246      * @param lineaABuscar
247      * @return
248      * @throws IOException
249      * @throws InterruptedException
250      */
251     @Override
252     public boolean esperarPantalla(String lineaABuscar)
253             throws IOException, InterruptedException {
254         Long maxTime = TIEMPO_EJEC_MAXIMO + System.currentTimeMillis();
255         String resultado = "";
256         do {
257             enviarComando(COMANDO_ASCII);
258             sleep(DELAY);
259             resultado = obtenerRespuestaMaquina();
260             if (resultado.contains(lineaABuscar)) {
261                 return true;
262             }
263 
264         } while (maxTime > System.currentTimeMillis());
265 
266         return false;
267     }
268 
269     /**
270      * Método para esperar para poder ejecutar el siguiente comando.
271      *
272      * @return
273      * @throws IOException
274      */
275     @Override
```

```
276     public boolean ejecutarSiguienteComando() throws IOException {
277         String line = "";
278         Long maxTime = TIEMPO_EJEC_MAXIMO + System.currentTimeMillis();
279         while (maxTime > System.currentTimeMillis()) {
280             line = inStream.readLine();
281             if (line == null ||
282                     line.matches(PATRON_RESPUESTA_MAINFRAME_ERROR)) {
283                 System.out.println(line);
284                 return false;
285             } else if (line.matches(PATRON_RESPUESTA_MAINFRAME_OK)) {
286                 return true;
287             }
288         }
289         return false;
290     }
291
292 }
293
```