```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       */
61      private boolean conectarHost(String host) throws IOException {
62          outStream.println(CONNECT + host);
63          outStream.flush();
64          return ejecutarSiguienteComando();
65      }
66
67      /**
68       * Obtiene la respuesta textual del mainframe en la terminal s3270.
69       *
70       * @return
71       * @throws IOException
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 (enviarNombreUsuario(username) &&
159                         enviarContrasenya(password)) {
160                     return true;
161                 }
162             }
163         }
164         return false;
165     }
```

```
166
167     /**
168      * Método auxiliar para enviar el nombre de usuario al emulador.
169      *
170      * @param username
171      * @return
172      * @throws IOException
173      * @throws InterruptedException
174      */
175     private boolean enviarNombreUsuario(String username)
176             throws IOException, InterruptedException {
177         if (esperarPantalla(PANTALLA_LOGIN)) {
178             if (enviarString(username)) {
179                 if (enviarComando(COMANDO_ENTER)) {
180                     return true;
181                 }
182             }
183         }
184         return false;
185     }
186
187     /**
188      * Método auxiliar para enviar la contraseña del usuario al emulador.
189      *
190      * @param password
191      * @return
192      * @throws IOException
193      */
194     private boolean enviarContrasenya(String password) throws IOException {
195         if (enviarString(password)) {
196             if (enviarComando(COMANDO_ENTER)) {
197                 return true;
198             }
199         }
200         return false;
201     }
202
203     /**
204      * Ejecuta la aplicación legada.
205      *
206      * @return
207      * @throws InterruptedException
208      * @throws IOException
209      */
210     public boolean ejecutarTasksJob() throws InterruptedException,
211             IOException {
212         if (esperarPantalla(PANTALLA_MENU_PRINCIPAL)) {
213             if (enviarString(TASKS2JOB)) {
214                 if (enviarComando(COMANDO_ENTER)) {
215                     if (esperarPantalla(MENU_TASKS2)) {
216                         return true;
217                     }
218                 }
219             }
220         }
```

```
221         return false;
222     }
223 
224     /**
225      * Realiza la salida controlada del mainframe a través de la terminal
226      * s3270.
227      *
228      * @return
229      * @throws IOException
230      */
231     @Override
232     public boolean logout() throws IOException {
233         enviarString(CADENA_OFF);
234         enviarComando(COMANDO_ENTER);
235         enviarComando(COMANDO_EXIT);
236 
237         return true;
238     }
239 
240     /**
241      * Método sobreescrito para enviar un comando al mainframe.
242      *
243      * @param comando
244      * @return
245      * @throws IOException
246      */
247     @Override
248     public boolean enviarComando(String comando) throws IOException {
249         outStream.println(comando);
250         outStream.flush();
251         if (comando.equals(COMANDO_ASCII)) {
252             return true;
253         } else {
254             return ejecutarSiguienteComando();
255         }
256     }
257 
258     /**
259      * Envía una cadena de texto a la terminal s3270.
260      *
261      * @param mensaje
262      * @return
263      * @throws IOException
264      */
265     @Override
266     public boolean enviarString(String mensaje) throws IOException {
267         outStream.println(String.format(FORMATO_CADENA_TEXTO, mensaje));
268         outStream.flush();
269         return ejecutarSiguienteComando();
270     }
271 
272     /**
273      * Método sobreecrito para esperar una pantalla del emulador.
274      *
275      * @param lineaABuscar
```

```java
     * @return
     * @throws IOException
     * @throws InterruptedException
     */
    @Override
    public boolean esperarPantalla(String lineaABuscar)
            throws IOException, InterruptedException {
        Long maxTime = TIEMPO_EJEC_MAXIMO + System.currentTimeMillis();
        String resultado = "";
        do {
            enviarComando(COMANDO_ASCII);
            sleep(DELAY);
            resultado = obtenerRespuestaMaquina();
            if (resultado.contains(lineaABuscar)) {
                return true;
            }

        } while (maxTime > System.currentTimeMillis());

        return false;
    }

    /**
     * Método para esperar para poder ejecutar el siguiente comando.
     *
     * @return
     * @throws IOException
     */
    @Override
    public boolean ejecutarSiguienteComando() throws IOException {
        String line = "";
        Long maxTime = TIEMPO_EJEC_MAXIMO + System.currentTimeMillis();
        while (maxTime > System.currentTimeMillis()) {
            line = inStream.readLine();
            if (line == null ||
                    line.matches(PATRON_RESPUESTA_MAINFRAME_ERROR)) {
                System.out.println(line);
                return false;
            } else if (line.matches(PATRON_RESPUESTA_MAINFRAME_OK)) {
                return true;
            }
        }
        return false;
    }
}
```