```java
/**
 * TasksJob.java
 * <p>
 * Clase que implementa los métodos para la ejecución de las
 * funciones de la aplicación legada.
 * <p>
 * Radu Constantin Robu y Alberto Pérez
 */
package P3.Control;

import P3.Modelo.Tarea;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class TasksJob implements TasksAPI {
    private static final String NUEVO_FICHERO = "N";
    private static final String GUARDAR = "s";
    private static final String ANYADIR = "a";
    private static final String LISTAR = "l";
    private static final String ELIMINAR = "r";
    private static final String BUSCAR = "t";
    private static final String SI = "y";

    private static final String PATRON_IDTAREA = "^data: TASK NUMBER: .*$";
    private static final String PATRON_NOMBRE = "^data: NAME.*$";
    private static final String PATRON_DESCRIPCION = "^data: DESCRIPTION.*$";
    private static final String PATRON_FECHA = "^data: DATE .*$";
    private static final String TEXTO_FECHA = "data: DATE        : ";
    private static final String TEXTO_DESCRIPCION = "data: DESCRIPTION: ";
    private static final String TEXTO_NOMBRE = "data: NAME        : ";
    private static final String TEXTO_ID_TAREA = "data: TASK NUMBER: ";

    private static final String MENU_TASKS2 = "**MENU**";
    private static final String MENSAJE_NUEVO_FICHERO = "**NEW TASK FILE**";
    private static final String FICHERO_TAREAS_CREADO =
            "NEW TASK FILE HAS BEEN CREATED";
    private static final String MENSAJE_SALIDA = "BYE";
    private static final String MENSAJE_GUARDAR_TAREAS = "SAVE TASKS";
    private static final String MENSAJE_LISTAR_TAREAS = "**LIST TASK**";
    private static final String TAREAS_GUARDADAS = "TASKS HAVE BEEN SAVED";
    private static final String MENSAJE_FIN_LISTA = "**END**";
    private static final String MENSAJE_BUSCAR_TAREAS = "**SEARCH TASK**";
    private static final String MENSAJE_CONFIRMAR_ELIMINAR = "CONFIRM (Y/N)";
    private static final String MENSAJE_TAREA_NO_ENCONTRADA = "TASK NOT FOUND";
    private static final String MENSAJE_ELIMINAR_TAREA = "**REMOVE TASK**";
    private static final String MENSAJE_ANYADIR_TAREA = "**ADD TASK**";

    private Mainframe mainframe;

    /**
     * Constructor de la clase.
     *
     * @param mainframe
```

```
56       * @throws IOException
57       */
58      public TasksJob(Mainframe mainframe) throws IOException {
59          this.mainframe = mainframe;
60      }
61
62      /**
63       * Opción de tasks2 para crear un nuevo fichero de tareas.
64       *
65       * @return
66       * @throws IOException
67       * @throws InterruptedException
68       */
69      @Override
70      public boolean nuevoFicheroTareas()
71              throws IOException, InterruptedException {
72          if (mainframe.enviarString(NUEVO_FICHERO)) {
73              if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
74                  if (mainframe.esperarPantalla(MENSAJE_NUEVO_FICHERO)) {
75                      if (mainframe.enviarString(SI)) {
76                          if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
77                              if (mainframe
78                                      .esperarPantalla(FICHERO_TAREAS_CREADO)) {
79                                  if (mainframe.enviarComando(
80                                          Mainframe.COMANDO_ENTER)) {
81                                      if (mainframe
82                                              .esperarPantalla(MENU_TASKS2)) {
83                                          return true;
84                                      }
85                                  }
86                              }
87                          }
88                      }
89                  }
90              }
91          }
92          return false;
93      }
94
95      /**
96       * Opción de tasks2 para añadir una nueva tarea.
97       *
98       * @param idTarea
99       * @param nombreTarea
100      * @param descripcionTarea
101      * @param fecha
102      * @return
103      * @throws IOException
104      * @throws InterruptedException
105      */
106     @Override
107     public CODIGO_ERROR anyadirTarea(String idTarea, String nombreTarea,
108                                      String descripcionTarea, String fecha)
109             throws IOException, InterruptedException {
110         CODIGO_ERROR codigo = CODIGO_ERROR.NOK;
```

```
111
112         if (existeIdTarea(idTarea)) {
113             return CODIGO_ERROR.IDTAREA_REPETIDO;
114         }
115
116         if (mainframe.enviarString(ANYADIR)) {
117             if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
118                 if (mainframe.esperarPantalla(MENSAJE_ANYADIR_TAREA)) {
119                     codigo = auxiliarEnviarDatosTarea(idTarea, nombreTarea,
120                             descripcionTarea, fecha);
121
122                 }
123             }
124         }
125         return codigo;
126     }
127
128     /**
129      * Método auxiliar para enviar los datos de la tarea.
130      *
131      * @param idTarea
132      * @param nombreTarea
133      * @param descripcionTarea
134      * @param fecha
135      * @return
136      * @throws IOException
137      * @throws InterruptedException
138      */
139     public CODIGO_ERROR auxiliarEnviarDatosTarea(
140             String idTarea, String nombreTarea, String descripcionTarea,
141             String fecha) throws IOException, InterruptedException {
142         if (mainframe.enviarString(idTarea)) {
143             if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
144                 if (mainframe.enviarString(nombreTarea)) {
145                     if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
146                         if (mainframe.enviarString(descripcionTarea)) {
147                             if (mainframe.enviarComando(
148                                     Mainframe.COMANDO_ENTER)) {
149                                 if (mainframe.enviarString(fecha)) {
150                                     if (mainframe.enviarComando(
151                                             Mainframe.COMANDO_ENTER)) {
152                                         if (mainframe.enviarComando(
153                                                 Mainframe.COMANDO_ENTER)) {
154                                             if (mainframe.esperarPantalla(
155                                                     MENU_TASKS2)) {
156                                                 return CODIGO_ERROR.OK;
157                                             }
158                                         }
159                                     }
160                                 }
161                             }
162                         }
163                     }
164                 }
165             }
```

```
166         }
167         return CODIGO_ERROR.NOK;
168     }
169 
170     /**
171      * Verifica si existe ya una tarea con el idTarea introducido.
172      *
173      * @param idTarea
174      * @return
175      * @throws IOException
176      * @throws InterruptedException
177      */
178     private boolean existeIdTarea(String idTarea)
179             throws IOException, InterruptedException {
180         List<Tarea> tareas = listarTareas();
181         for (Tarea t : tareas) {
182             if (t.getId().equals(idTarea)) {
183                 return true;
184             }
185         }
186         return false;
187     }
188 
189     /**
190      * Opción de tasks2 para eliminar una tarea.
191      *
192      * @param idTarea
193      * @return
194      * @throws IOException
195      * @throws InterruptedException
196      */
197     @Override
198     public CODIGO_ERROR eliminarTarea(String idTarea)
199             throws IOException, InterruptedException {
200         CODIGO_ERROR codigo = CODIGO_ERROR.NOK;
201         if (mainframe.enviarString(ELIMINAR)) {
202             if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
203                 if (mainframe.esperarPantalla(MENSAJE_ELIMINAR_TAREA)) {
204                     if (mainframe.enviarString(idTarea)) {
205                         if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
206                             codigo = auxiliarEliminarTarea();
207                         }
208                     }
209                 }
210             }
211         }
212         return codigo;
213     }
214 
215     /**
216      * Método auxiliar para eliminar la tarea.
217      *
218      * @param idTarea
219      * @return
220      * @throws IOException
```

```
221      * @throws InterruptedException
222      */
223     public CODIGO_ERROR auxiliarEliminarTarea()
224             throws IOException, InterruptedException {
225         if (mainframe.esperarPantalla(MENSAJE_TAREA_NO_ENCONTRADA)) {
226             if (mainframe.enviarComando(MainframeAPI.COMANDO_ENTER)) {
227                 if (mainframe.esperarPantalla(MENU_TASKS2)) {
228                     return CODIGO_ERROR.IDTAREA_INCORRECTO;
229                 }
230             }
231         } else if (mainframe.esperarPantalla(MENSAJE_CONFIRMAR_ELIMINAR)) {
232             if (mainframe.enviarString(SI)) {
233                 if (mainframe.enviarComando(MainframeAPI.COMANDO_ENTER)) {
234                     if (mainframe.enviarComando(MainframeAPI.COMANDO_ENTER)) {
235                         if (mainframe.esperarPantalla(MENU_TASKS2)) {
236                             return CODIGO_ERROR.OK;
237                         }
238                     }
239                 }
240             }
241         }
242         return CODIGO_ERROR.NOK;
243     }
244 
245     /**
246      * Opción de tasks2 para buscar las tareas de una fecha concreta.
247      *
248      * @param fecha
249      * @return
250      * @throws IOException
251      * @throws InterruptedException
252      */
253     @Override
254     public List<Tarea> buscarTareas(String fecha)
255             throws IOException, InterruptedException {
256         List<Tarea> tareas = new ArrayList();
257         if (mainframe.enviarString(BUSCAR)) {
258             if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
259                 if (mainframe.esperarPantalla(MENSAJE_BUSCAR_TAREAS)) {
260                     if (mainframe.enviarString(fecha)) {
261                         if (mainframe.enviarComando(
262                                 Mainframe.COMANDO_ENTER)) {
263                             if (mainframe.esperarPantalla(MENSAJE_FIN_LISTA)) {
264                                 if (mainframe.enviarComando(
265                                         Mainframe.COMANDO_ASCII)) {
266                                     tareas = obtenerListaTareas();
267                                     if (mainframe.enviarComando(
268                                             Mainframe.COMANDO_ENTER)) {
269                                         if (mainframe.esperarPantalla(
270                                                 MENU_TASKS2)) {
271                                             return tareas;
272                                         }
273                                     }
274                                 }
275                             }
```

```
                        }
                    }
                }
            }
        }
        return tareas;
    }

    /**
     * Opción de tasks2 para listar las tareas.
     *
     * @return
     * @throws IOException
     * @throws InterruptedException
     */
    @Override
    public List<Tarea> listarTareas()
            throws IOException, InterruptedException {
        List<Tarea> tareas = new ArrayList();
        if (mainframe.enviarString(LISTAR)) {
            if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
                if (mainframe.esperarPantalla(MENSAJE_LISTAR_TAREAS)) {
                    if (mainframe.enviarComando(Mainframe.COMANDO_ASCII)) {
                        tareas = obtenerListaTareas();
                        if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
                            if (mainframe.esperarPantalla(MENU_TASKS2)) {
                                return tareas;
                            }
                        }
                    }
                }
            }
        }
        return tareas;
    }

    /**
     * Obtiene la lista de tareas.
     *
     * @return
     * @throws IOException
     */
    private List<Tarea> obtenerListaTareas() throws IOException {
        String resultado = mainframe.obtenerRespuestaMaquina();
        String[] lineas = resultado.split("\n");
        return parsearTareas(lineas);
    }

    /**
     * Parsea la lista de tareas recibida del mainframe.
     *
     * @param resultado
     * @return
     */
    private List<Tarea> parsearTareas(String[] resultado) {
```

```
331         String idTarea = "";
332         String nombre = "";
333         String descripcion = "";
334         String fecha = "";
335         List<Tarea> tareas = new ArrayList();
336 
337         for (String line : resultado) {
338             if (line.matches(PATRON_IDTAREA)) {
339                 idTarea = line.replace(TEXTO_ID_TAREA, "").strip();
340             } else if (line.matches(PATRON_NOMBRE)) {
341                 nombre = line.replace(TEXTO_NOMBRE, "").strip();
342             } else if (line.matches(PATRON_DESCRIPCION)) {
343                 descripcion = line.replace(TEXTO_DESCRIPCION, "").strip();
344             } else if (line.matches(PATRON_FECHA)) {
345                 fecha = line.replace(TEXTO_FECHA, "").strip();
346                 tareas.add(new Tarea(idTarea, nombre, descripcion, fecha));
347             }
348         }
349         return tareas;
350     }
351 
352     /**
353      * Opción de tasks2 para guardar las tareas.
354      *
355      * @return
356      * @throws IOException
357      * @throws InterruptedException
358      */
359     @Override
360     public boolean guardarTareas()
361             throws IOException, InterruptedException {
362         if (mainframe.enviarString(GUARDAR)) {
363             if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
364                 if (mainframe.esperarPantalla(TAREAS_GUARDADAS)) {
365                     if (mainframe.enviarComando(Mainframe.COMANDO_ENTER)) {
366                         if (mainframe.esperarPantalla(MENU_TASKS2)) {
367                             return true;
368                         }
369                     }
370                 }
371             }
372         }
373         return false;
374     }
375 
376     /**
377      * Termina la comunicación con la terminal s3270.
378      *
379      * @param guardarTareas
380      * @return
381      * @throws IOException
382      * @throws InterruptedException
383      */
384     @Override
385     public boolean salir(String guardarTareas)
```

```
386             throws IOException, InterruptedException {
387         if (mainframe.enviarString(Mainframe.COMANDO_EXIT)) {
388             if (mainframe.enviarComando(MainframeAPI.COMANDO_ENTER)) {
389                 if (mainframe.esperarPantalla(MENSAJE_SALIDA)) {
390                     if (mainframe.enviarComando(MainframeAPI.COMANDO_ENTER)) {
391                         return true;
392                     }
393                 } else if (mainframe.esperarPantalla(MENSAJE_GUARDAR_TAREAS)) {
394                     if (mainframe.enviarString(guardarTareas)) {
395                         if (mainframe.enviarComando(
396                                 MainframeAPI.COMANDO_ENTER)) {
397                             if (mainframe.esperarPantalla(MENSAJE_SALIDA)) {
398                                 if (mainframe.enviarComando(
399                                         MainframeAPI.COMANDO_ENTER)) {
400                                     return true;
401                                 }
402                             }
403                         }
404                     }
405                 }
406             }
407         }
408         return false;
409     }
410 }
411
```