View Javadoc

```
1   package model.element.mobile;
2
3   import java.awt.Image;
4   import java.io.IOException;
5
6   import javax.imageio.ImageIO;
7
8   import contract.ElementType;
9   import contract.IElement;
10  import contract.ILevelMap;
11  import model.element.LevelMap;
12
13  /**
14   * The Class Diamond.
15   *
16   * @author Group 5
17   */
18  public class Player implements IElement {
19
20      /** The score */
21      private int score = 0;
22
23      /** The exist boolean */
24      private boolean exist = true;
25
26      /** The elementType */
27      private ElementType elementType = ElementType.PLAYER;
28
29      /** The x */
30      private int x;
31
32      /** The y */
33      private int y;
34
35      /** The levelmap */
36      private ILevelMap levelmap;
37
38      /** The image */
39      private static Image image;
40
41      /** The image up */
42      private String imageNameUp = "JoueurMonteArret";
43
44      /** The image down */
45      private String imageNameDown = "JoueurDescendArret";
46
47      /** The image right */
48      private String imageNameRight = "JoueurDroiteArret";
49
50      /** The image left */
51      private String imageNameLeft = "JoueurGaucheArret";
52
53      /** The imageName */
54      private String imageName;
55
56      /**
57       * constructor to build and place player
58       *
59       * @param x
60       *              The x.
61       * @param y
62       *              The y.
63       * @param levelMap
64       *              The levelMap.
65       *
66       */
67      public Player(final int x, final int y, LevelMap levelMap) {
68              this.setX(x);
69              this.setY(y);
70              this.setImageName(imageNameDown);
71              this.loadImage();
72              this.setLevelmap(levelMap);
73
74      }
75
76      /**
77       * Get x position of player
78       *
79       * @return x
80       */
81      @Override
82      public int getX() {
83
84              return this.x;
85      }
86
87      /**
88       * Set x position of player
89       *
90       * @param x
91       *              The x.
```

```
92       */
93      @Override
94      public void setX(int x) {
95
96              this.x = x;
97      }
98
99      /**
100      * Get y position of player
101      *
102      * @return y
103      */
104     @Override
105     public int getY() {
106
107             return this.y;
108     }
109
110     /**
111      * Set y position of player
112      *
113      * @param y
114      *              The y.
115      */
116     @Override
117     public void setY(int y) {
118
119             this.y = y;
120     }
121
122     /**
123      * Move up the player when up key are pressed
124      *
125      */
126     @Override
127     public void moveUp() {
128
129             this.setY(this.getY() - 1);
130             this.setImageName(imageNameUp);
131             this.loadImage();
132
133             this.levelmap.setElement(this.getX(), this.getY(), this);
134             this.levelmap.removeElement(getX(), getY()+1);
135
136     }
137
138     /**
139      * Move down the player when down key are pressed
140      *
141      */
142     @Override
143     public void moveDown() {
144             this.setY(this.getY() + 1);
145             this.setImageName(imageNameDown);
146             this.loadImage();
147
148             this.levelmap.setElement(this.getX(), this.getY(), this);
149             this.levelmap.removeElement(getX(), getY()-1);
150     }
151
152     /**
153      * Move left the player when left key are pressed
154      *
155      */
156     @Override
157     public void moveLeft() {
158             this.setX(this.getX() - 1);
159             this.setImageName(imageNameLeft);
160             this.loadImage();
161
162             this.levelmap.setElement(this.getX(), this.getY(), this);
163             this.levelmap.removeElement(getX()+1, getY());
164     }
165
166     /**
167      * Move right the player when right key are pressed
168      *
169      */
170     @Override
171     public void moveRight() {
172             this.setX(this.getX() + 1);
173             this.setImageName(imageNameRight);
174             this.loadImage();
175
176             this.levelmap.setElement(this.getX(), this.getY(), this);
177             this.levelmap.removeElement(getX()-1, getY());
178     }
179
180     /**
181      * do nothing the player when player don't move
182      *
183      */
184     @Override
```

```
185     public void doNothing() {
186             this.setY(this.getY());
187             this.setImageName(imageNameDown);
188             this.loadImage();
189 
190             this.levelmap.setElement(this.getX(), this.getY(), this);
191     }
192 
193     /**
194      * Get image of player
195      *
196      * @return image
197      */
198     @Override
199     public Image getImage() {
200 
201             return Player.image;
202     }
203 
204     /**
205      * Set image of player
206      *
207      * @param image
208      *                      The image.
209      */
210     @Override
211     public void setImage(Image image) {
212 
213             Player.image = image;
214     }
215 
216     /**
217      * Load image of player
218      *
219      */
220     @Override
221     public void loadImage() {
222 
223             Image img = null;
224             try {
225                     img = ImageIO.read(getClass().getClassLoader().getResourceAsStream("images/" + this.getImageName() + ".png"));
226             }
227             catch(IOException e) {
228                     e.printStackTrace();
229             }
230             this.setImage(img);
231     }
232 
233     /**
234      * Get image name of player
235      *
236      * @return imgaName
237      *
238      */
239     @Override
240     public String getImageName() {
241 
242             return this.imageName;
243     }
244 
245     /**
246      * Set image name of player
247      *
248      * @param imageName
249      *                              The image name.
250      */
251     @Override
252     public void setImageName(String imageName) {
253 
254             this.imageName = imageName;
255     }
256 
257 
258     /**
259      * check existing of player
260      *
261      * @return exist
262      *
263      */
264     @Override
265     public boolean isExist() {
266 
267             return this.exist;
268     }
269 
270     /**
271      * set exist verification of player
272      *
273      * @param exist
274      *                      The exist state.
275      *
276      */
277     @Override
```

```
278     public void setExist(boolean exist) {
279             this.exist = exist;
280     }
281
282     /**
283      * Get level
284      *
285      * @return level map
286      *
287      */
288     public ILevelMap getLevelmap() {
289             return levelmap;
290     }
291
292     /**
293      * Set level
294      *
295      * @param levelmap
296      *                      The levelMap.
297      *
298      */
299     public void setLevelmap(ILevelMap levelmap) {
300             this.levelmap = levelmap;
301     }
302
303     /**
304      * Get score of collected diamond
305      *
306      * @return score
307      *
308      */
309     @Override
310     public int getScore() {
311             return score;
312     }
313
314     /**
315      * Set score of collected diamond
316      *
317      * @param score
318      *                      The score.
319      *
320      */
321     @Override
322     public void setScore(int score) {
323             this.score = score;
324     }
325
326     /**
327      * Get element type of player
328      *
329      * @return element type
330      *
331      */
332     @Override
333     public ElementType getElementType() {
334             return elementType;
335     }
336
337     /**
338      * Set element type of player
339      *
340      * @param elementType
341      *                              The elementType.
342      *
343      */
344     @Override
345     public void setElementType(ElementType elementType) {
346             this.elementType = elementType;
347     }
348
349 }
```