View Javadoc

```
1   package model.element.motionless;
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 Exit 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.EXIT;
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 imageName */
42      private String imageName = "trapdoor";
43
44      /**
45       * constructor to build and place Exit
46       *
47       * @param x
48       *              The x.
49       * @param y
50       *              The y.
51       * @param levelMap
52       *              The levelMap.
53       *
54       */
55      public Exit(final int x, final int y, LevelMap levelMap) {
56              this.setX(x);
57              this.setY(y);
58              this.setImageName(imageName);
59              this.loadImage();
60              this.setLevelmap(levelMap);
61
62      }
63
64      /**
65       * Get x position of Exit
66       *
67       * @return x
68       *
69       */
70      @Override
71      public int getX() {
72
73              return this.x;
74
75      }
76
77      /**
78       * Set x position of Exit
79       *
80       * @param x
81       *              The x.
82       *
83       */
84      @Override
85      public void setX(int x) {
86
87              this.x = x;
88
89      }
90
91      /**
```

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

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

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