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 Enemy.
15   *
16   * @author Group 5
17   */
18  public class Enemy 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.ENEMY;
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 = "BatMonteArret";
43
44      /** The image down */
45      private String imageNameDown = "BatDescendArret";
46
47      /** The image right */
48      private String imageNameRight = "BatDroiteArret";
49
50      /** The image left */
51      private String imageNameLeft = "BatGaucheArret";
52
53      /** The imageName */
54      private String imageName;
55
56      /**
57       * constructor to build and place enemy
58       *
59       * @param x
60       *              The x.
61       * @param y
62       *              The y.
63       * @param levelMap
64       *              The levelMap.
65       *
66       */
67      public Enemy(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 Enemy
78       *
79       * @return x
80       */
81      @Override
82      public int getX() {
83
84              return this.x;
85      }
86
87      /**
88       * Set x position of Enemy
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 Enemy
101      *
102      * @return y
103      */
104     @Override
105     public int getY() {
106
107             return this.y;
108     }
109
110     /**
111      * Set y position of Enemy
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     @Override
126     public void moveUp() {
127
128             this.setY(this.getY() - 1);
129             this.setImageName(imageNameUp);
130             this.loadImage();
131
132             this.levelmap.setElement(this.getX(), this.getY(), this);
133             this.levelmap.removeElement(getX(), getY()+1);
134
135     }
136
137     /**
138      * Move down the player when down key are pressed
139      */
140     @Override
141     public void moveDown() {
142             this.setY(this.getY() + 1);
143             this.setImageName(imageNameDown);
144             this.loadImage();
145
146             this.levelmap.setElement(this.getX(), this.getY(), this);
147             this.levelmap.removeElement(getX(), getY()-1);
148     }
149
150     /**
151      * Move left the player when left key are pressed
152      */
153     @Override
154     public void moveLeft() {
155             this.setX(this.getX() - 1);
156             this.setImageName(imageNameLeft);
157             this.loadImage();
158
159             this.levelmap.setElement(this.getX(), this.getY(), this);
160             this.levelmap.removeElement(getX()+1, getY());
161     }
162
163     /**
164      * Move right the player when right key are pressed
165      */
166     @Override
167     public void moveRight() {
168             this.setX(this.getX() + 1);
169             this.setImageName(imageNameRight);
170             this.loadImage();
171
172             this.levelmap.setElement(this.getX(), this.getY(), this);
173             this.levelmap.removeElement(getX()-1, getY());
174     }
175
176     /**
177      * do nothing the player when player don't move
178      */
179     @Override
180     public void doNothing() {
181             this.setY(this.getY());
182             this.setImageName(imageNameDown);
183             this.loadImage();
184
```

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

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