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 UnbreakableBlock 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.UNBREAKABLEBLOCK;
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 = "bedrock";
43
44      /**
45       * constructor to build and place UnbreakableBlock
46       *
47       * @param x
48       *              The x.
49       * @param y
50       *              The y.
51       * @param levelMap
52       *              The levelMap
53       *
54       */
55      public UnbreakableBlock(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 UnbreakableBlock
66       *
67       * @return x
68       *
69       */
70      @Override
71      public int getX() {
72
73              return this.x;
74      }
75
76      /**
77       * Set x position of UnbreakableBlock
78       *
79       * @param x
80       *              The x.
81       *
82       */
83      @Override
84      public void setX(int x) {
85
86              this.x = x;
87      }
88
89      /**
90       * Get y position of UnbreakableBlock
91       *
```

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

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

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