Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Программирование

Отчет по курсовой работе
Игра "Побег из гробницы"

Работу выполнил:
Бойцов К.С.
Группа:
23501/4
Преподаватель:
Вылегжанина К.Д.

Санкт-Петербург
2017

Содержание

1 Игровое приложение: ”Побег из гробницы”

Главный герой игры - археолог - оказывается в одном помещении с ожившей мумией, желающей расправиться с незваным гостем.

1.1 Концепция игрового приложения ”Побег из гробницы”

Приложение являет собой логическую игру, в которой главному герою предстоит добраться до выхода из гробницы.

Приложение отрисовывает игровое поле, на которое помещаются два существа. Пользователь может управлять главным героем с помощью нажатия на клавиши-стрелки. Мумией управляет искусственный интеллект, заставляющий её двигаться по направлению к герою. Также на поле размещены преграды, мешающие существам пройти. Игра считается законченой, если пользователь довёл протагониста до выхода и не дал мумии догнать его.
Игровое поле составляется вне приложения в текстовом файле.

1.2 Задание

Разработать приложение под операционные системы Windows 7+ и Android, позволяющее играть в "Побег из гробницы".

1.3 Минимально работоспособный продукт

Приложение, которое предоставляет возможность передвигать главного героя.

1.4 Вывод

Пояснён выбор темы курсового проекта. Описана концепция игры ”Побег из гробницы”. Определено задание.

2 Проектирование графического приложения, реализующего игру ”Побег из гробницы”

2.1 Проектирование библиотеки

Приложение должно позволять игроку играть в ”Побег из гробницы”

Библиотека - ядро приложения. Здесь содержатся основные классы, необходимые для представления игры. Для создания графического приложения была выбрана библиотека LibGDX

3 Реализация игры ”Побег из гробницы”

3.1 Среда разработки

Интегрированная среда разработки IntelliJ IDEA 2016.2.5.
Язык: Java 1.8.
Система автоматической сборки: Gradle 2.14.

3.2 Реализация библиотеки

Классы библиотеки объединены в пакет logic. Основные классы, выделенные в библиотеке:

1 Класс Field. Содержит в себе игровое поле. Хранит массив из клеток, имеющих своё состояние.

2 Класс Cell. Клетка. Составляющая часть поля, доступно три состояния: свободная клетка, заблокированная клетка, выход из гробницы.

3 Класс Player. Хранит существ и хранит их координаты. Отвечает за их передвижение.

3.3 Реализация графического приложения

Для создания графического приложения была выбрана библиотека Swing.

На рисунке 1 представлен вариант игрового поля для игры ”Побег из гробницы”, где 0 - свободная клетка, 1 - преграда, 2 - выход:

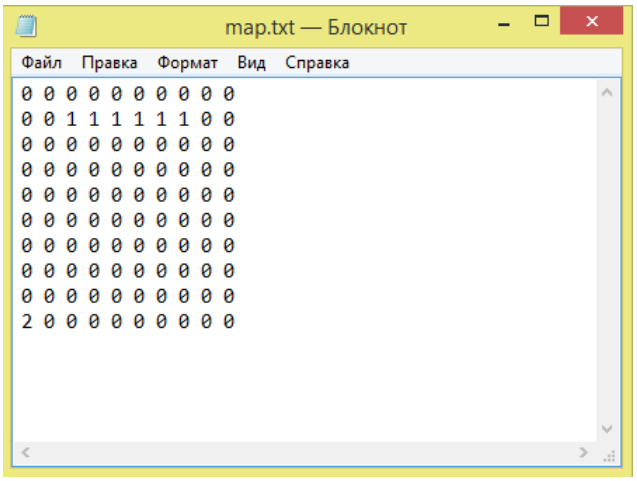

Рис. 1: Вручную созданное поле для игры.

На рисунке 2 изображено соответствующая ему графическая интерпретация. Положение главного героя и мумии задано пользователем. На дан-

ный момент преграды и выход никак не обозначены, что будет исправлено в скорейшем времени.

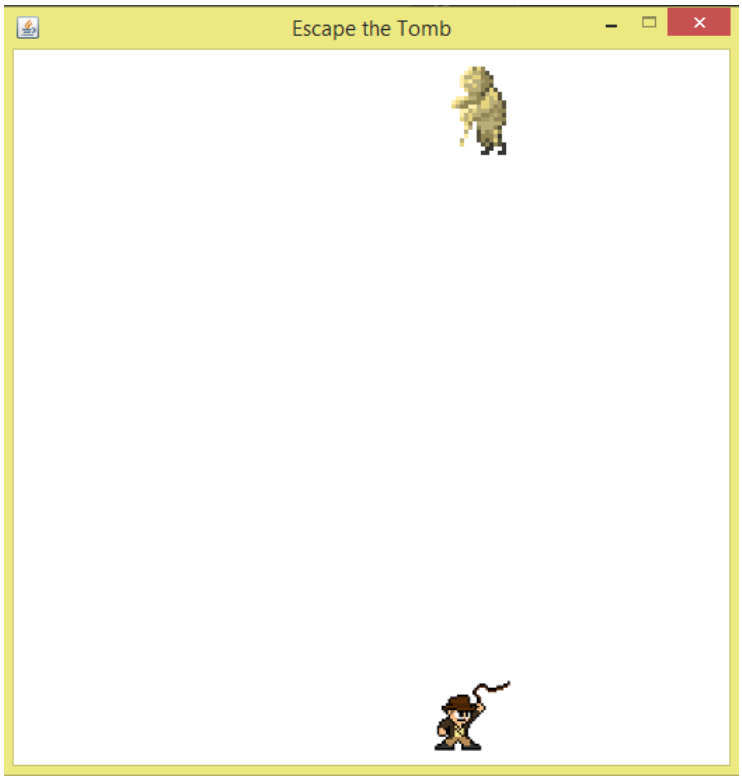

Рис. 2: Игровое поле с расположенными на нём игровыми персонажами

Игра заканчивается в двух случаях: либо при достижении археологом выхода, либо при его столкновении с мумией.

Мумия заблокирована преградой, поэтому игрок может сделать к мумии несколько шагов, не боясь проиграть, как на рисунке 3:

Рис. 3: Мумия не может пройти к игроку

Победа будет засчитана, когда персонаж доберётся до выхода, как на рисунке 4. В данном случае выход находится в левом нижнем углу поля.

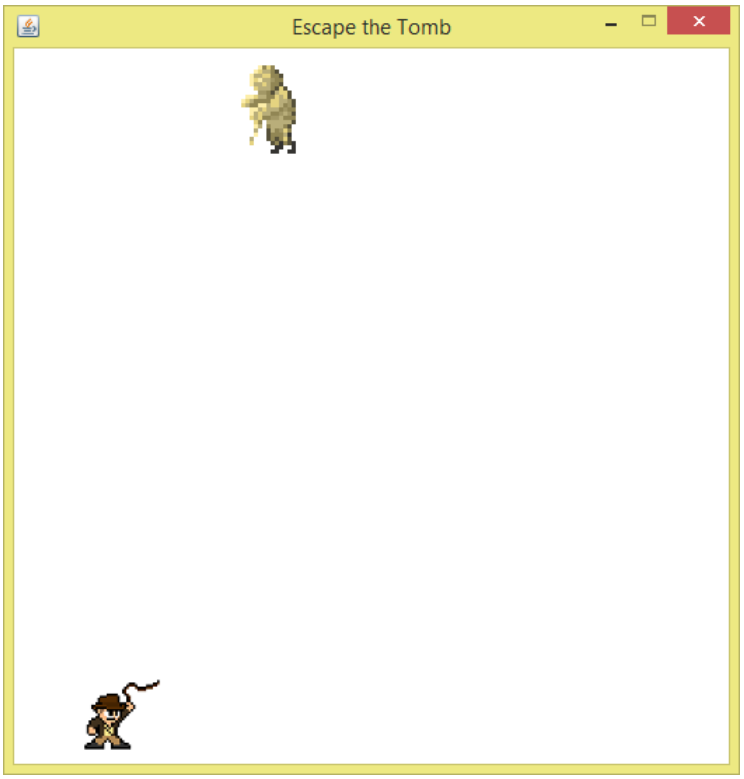

Рис. 4: Игрок добрался до выхода

Также игрок может быть пойман мумией, в результате чего он проиграет, как на рисунке 5.

Рис. 5: Декорирование главного экрана

При обоих исходах будет выведена на экран надпись Game over. Рисунок 6:

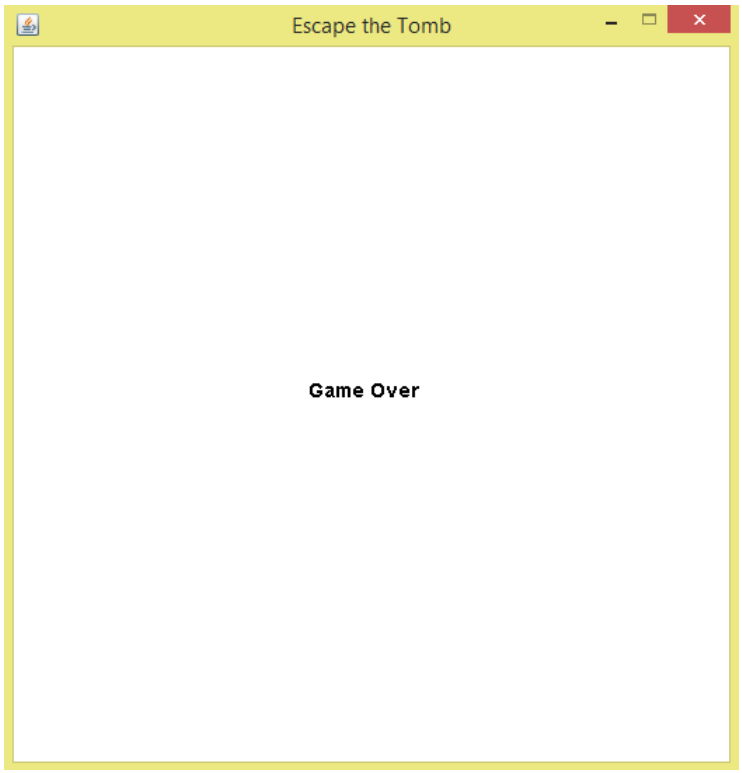

Рис. 6: Игра окончена

Таким образом разработано графическое приложение, позволяющее играть в ”Побег из гробницы”.

3.4 Вывод

Для реализации игры определены основные классы библиотеки, графического приложения. Разработано графическое приложение, предоставляющее возможность играть одному игроку.

4 Вывод

В результате работы было разработано игровое приложение ”Побег из гробницы”. Посредством библиотеки Swing было создано приложение, предоставляющее пользователю функциональность ядра и позволяющее играть в игру ”Побег из гробницы”.

5 Приложение

5.1 Листинги

```
1  package logic;
2
3  /**
4   * Created by Константин on 05.03.2017.
5   */
6  public class Cell {
7      private int x;
8      private int y;
9      private boolean isBlocked;
10     private boolean isExit;
11
12     public Cell(int x, int y, boolean isBlocked, boolean isExit) {
13         this.x = x;
14         this.y = y;
15         this.isBlocked = isBlocked;
16         this.isExit = isExit;
17     }
18     public int getX() {
19         return x;
20     }
21     public int getY() {
22         return y;
23     }
24     public boolean getIsBlocked() {
25         return isBlocked;
26     }
27     public boolean getIsExit() {
28         return isExit;
29     }
30 }
```

```
1  package logic;
2
3  import java.awt.*;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.util.Scanner;
7
8  /**
9   * Created by Константин on 05.03.2017.
10  */
11 public class Field {
12     private int x;
13     private int y;
14     private int field[][] = new int[10][10];
15     private Cell[][] cells = new Cell[10][10];
16
17     public int getHorizontal() {
18         return field.length;
19     }
20     public int getVertical() {
21         return field[0].length;
22     }
23
24     public Field (String path) {
25         try {
26             Scanner sc = new Scanner(new File(path));
```

```
27              for (int j = 0; j<10; j++) {
28                  for (int i = 0; i<10; i++) {
29                      field[i][j] = sc.nextInt();
30
31                      if (field[i][j] == 0) {
32                          cells[i][j] = new Cell(i,j,false,false);
33                      }
34                      if (field[i][j] == 1) {
35                          cells[i][j] = new Cell(i,j,true,false);
36                      }
37                      if (field[i][j] == 2) {
38                          cells[i][j] = new Cell(i,j,false,true);
39                      }
40                  }
41              }
42          }catch (FileNotFoundException exException) {
43              System.out.println("File not found");
44          }
45      }
46
47      public int[][] getField() {
48          return field;
49      }
50
51      public boolean ifPossibleToMove(int x,int y) {
52          return (!cells[x][y].getIsBlocked());
53      }
54
55  //      public void drawBoard(Graphics g) {
56  //          for (int j = 0; j < 500; j+=50) {
57  //              for (int i = 0; i < 500; i+=50) {
58  //                  if (field[i/50][j/50]==0) {
59  //                      g.setColor(Color.WHITE);
60  //                      g.fillRect(50,50,50,50);
61  //                  }
62  //                  if (field[i/50][j/50]==1) {
63  //                      g.setColor(Color.BLACK);
64  //                      g.fillRect(50,50,50,50);
65  //                  }
66  //                  if (field[i/50][j/50]==2) {
67  //                      g.setColor(Color.YELLOW);
68  //                      g.fillRect(50,50,50,50);
69  //                  }
70  //              }
71  //          }
72  //      }
73  }
```

```
1  package logic;
2
3  import GUI.Sprite;
4
5  import java.awt.*;
6  import javax.swing.ImageIcon;
7
8  /**
9   * Created by Константин on 05.03.2017.
10  */
11 public class Player extends Sprite {
12
13
14      private Image image;
```

```
15
16 //     public int getX() {
17 //         return x;
18 //     }
19 //
20 //     public int getY() {
21 //         return y;
22 //     }
23
24 //     public Player(int x,int y,Image image) {
25 //         this.x = x;
26 //         this.y = y;
27 //         this.image = image;
28 //     }
29
30     public Player(int x, int y, String path) {
31         super(x, y);
32
33         initPlayer(path);
34     }
35
36     private void initPlayer(String path) {
37
38         loadImage(path);
39         getImageDimensions();
40     }
41
42 //     public int getWidth() {
43 //         return image.getWidth(null);
44 //     }
45 //
46 //     public int getHeight() {
47 //         return image.getHeight(null);
48 //     }
49
50     public void draw(Graphics g,int x,int y) {
51         g.drawImage(image,x,y,null);
52     }
53
54 //     public Image getImage() {
55 //         return image;
56 //     }
57
58     public void moveX(int x1) {
59         x += x1;
60     }
61
62     public void moveY(int y1) {
63         y += y1;
64     }
65
66 }
```

```
1 package GUI;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 /**
7  * Created by Константин on 05.03.2017.
8  */
9
```

```
10 public class Sprite {
11
12     protected int x;
13     protected int y;
14     protected int width;
15     protected int height;
16     protected boolean vis;
17     protected Image image;
18
19     public Sprite(int x, int y) {
20
21         this.x = x;
22         this.y = y;
23         vis = true;
24     }
25
26     protected void getImageDimensions() {
27
28         width = image.getWidth(null);
29         height = image.getHeight(null);
30     }
31
32     protected void loadImage(String imageName) {
33
34         ImageIcon ii = new ImageIcon(imageName);
35         image = ii.getImage();
36     }
37
38     public Image getImage() {
39         return image;
40     }
41
42     public int getX() {
43         return x;
44     }
45
46     public int getY() {
47         return y;
48     }
49
50     public boolean isVisible() {
51         return vis;
52     }
53
54     public void setVisible(Boolean visible) {
55         vis = visible;
56     }
57
58     public Rectangle getBounds() {
59         return new Rectangle(x, y, width, height);
60     }
61 }
```

```
1 package GUI;
2
3 import logic.Field;
4 import logic.Player;
5
6 import javax.swing.*;
7 import java.awt.*;
8 import java.awt.event.KeyAdapter;
9 import java.awt.event.KeyEvent;
```

```
10 import java.awt.image.BufferStrategy;
11
12 import java.awt.BorderLayout;
13 import java.awt.Canvas;
14 import java.awt.Color;
15 import java.awt.Dimension;
16 import java.awt.Graphics;
17 import java.awt.Toolkit;
18 import java.awt.image.BufferStrategy;
19 import java.awt.image.BufferedImage;
20 import java.io.IOException;
21 import java.net.URL;
22
23 import javax.imageio.ImageIO;
24 import javax.swing.JFrame;
25
26
27 /**
28  * Created by Константин on 05.03.2017.
29  */
30 public class Game extends JPanel implements Runnable {
31
32     public boolean running;
33
34
35     private int x = 300;
36     private int y = 450;
37     private int mobX = 300;
38     private int mobY = 0;
39
40     ImageIcon ii = new ImageIcon("assets/hero.png");
41     private Image image = ii.getImage();
42
43     public Player hero = new Player(x,y,"assets/hero.png");
44     public Player mob = new Player(mobX,mobY,"assets/mummy.png");
45
46     String path = "src/maps/map.txt";
47     Field playfield = new Field(path);
48
49
50     private boolean leftPressed = false;
51     private boolean rightPressed = false;
52     private boolean downPressed = false;
53     private boolean upPressed = false;
54
55
56     public static int WIDTH = 500;
57     public static int HEIGHT = 500;
58     public static String NAME = "Escape the Tomb";
59
60     public void run(){
61         while(running) {
62             try {
63                 Thread.sleep(50);
64             } catch (InterruptedException e) {
65                 System.err.println(e);
66             }
67
68             update();
69         }
70     }
71
```

```
72      public void update(){
73          if (leftPressed && playfield.ifPossibleToMove((hero.getX()
        ↪ -50)/50,hero.getY()/50)) {
74              isGameOver();
75              hero.moveX(-50);
76              isWin();
77              moveForMob();
78          }
79          if (rightPressed && playfield.ifPossibleToMove((hero.getX()
        ↪ +50)/50,hero.getY()/50)) {
80              isGameOver();
81              hero.moveX(50);
82              isWin();
83              moveForMob();
84          }
85          if (downPressed && playfield.ifPossibleToMove(hero.getX()
        ↪ /50,(hero.getY()+50)/50)) {
86              isGameOver();
87              hero.moveY(50);
88              isWin();
89              moveForMob();
90          }
91          if (upPressed && playfield.ifPossibleToMove(hero.getX()
        ↪ /50,(hero.getY()-50)/50)) {
92              isGameOver();
93              hero.moveY(-50);
94              isWin();
95              moveForMob();
96          }
97
98          repaint();
99      }
100
101     public void init(){
102         setFocusable(true);
103         addKeyListener(keyListener);
104 //          addKeyListener(new TAdapter());
105
106         setBackground(Color.WHITE);
107     }
108
109 //    public void render(){
110 //        BufferStrategy bs = getBufferStrategy();
111 //        if (bs == null){
112 //            createBufferStrategy(2);
113 //            requestFocus();
114 //            return;
115 //        }
116 //
117 //        Graphics g = bs.getDrawGraphics();
118 //        g.setColor(Color.black);
119 //        g.fillRect(0, 0, getWidth(), getHeight());
120 //        hero.draw(g, x, y);
121 //        mob.draw(g, mobX, mobY);
122 //        g.dispose();
123 //        bs.show();
124 //    }
125
126     Graphics g;
127     public void start(){
128         running = true;
129
```

```
130         new Thread(this).start();
131     }
132 
133 
134 
135     @Override
136     public void paintComponent(Graphics g) {
137         super.paintComponent(g);
138 
139         if (running) {
140 
141             drawObjects(g);
142 
143 
144         } else {
145 
146             drawGameOver(g);
147         }
148 
149         Toolkit.getDefaultToolkit().sync();
150     }
151 
152     private void drawObjects(Graphics g) {
153 
154         if (hero.isVisible()) {
155             g.drawImage(hero.getImage(), hero.getX(), hero.getY(),
    ↪ this);
156         }
157 
158 
159         if (mob.isVisible()) {
160             g.drawImage(mob.getImage(), mob.getX(), mob.getY(),
    ↪ this);
161         }
162 
163 
164     }
165 
166     private void drawGameOver(Graphics g) {
167 
168         String msg = "Game Over";
169         Font small = new Font("Helvetica", Font.BOLD, 14);
170         FontMetrics fm = getFontMetrics(small);
171 
172         g.setColor(Color.black);
173         g.setFont(small);
174         g.drawString(msg, (WIDTH - fm.stringWidth(msg)) / 2, HEIGHT
    ↪ / 2);
175     }
176     private void drawCongratulations(Graphics g) {
177 
178         String msg = "You won!";
179         Font small = new Font("Helvetica", Font.BOLD, 14);
180         FontMetrics fm = getFontMetrics(small);
181 
182         g.setColor(Color.white);
183         g.setFont(small);
184         g.drawString(msg, (WIDTH - fm.stringWidth(msg)) / 2, HEIGHT
    ↪ / 2);
185     }
186 
187     private void drawDoor(Graphics gr) {
```

```
188             gr.setColor(Color.CYAN);
189             gr.fillRect(0,450,50,50);
190         }
191
192
193         public static void main(String[] args) {
194             Game game = new Game();
195             game.setPreferredSize(new Dimension(WIDTH, HEIGHT));
196
197             JFrame frame = new JFrame(Game.NAME);
198             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
199             frame.setLayout(new BorderLayout());
200             frame.add(game, BorderLayout.CENTER);
201             frame.pack();
202             frame.setResizable(false);
203             frame.setVisible(true);
204
205             game.init();
206 //              game.render();
207             game.start();
208         }
209
210
211         private KeyAdapter keyListener = new KeyAdapter() {
212             @Override
213             public void keyPressed(KeyEvent e) {
214                 if (e.getKeyCode() == KeyEvent.VK_LEFT) {
215                     leftPressed = true;
216                 }
217                 if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
218                     rightPressed = true;
219                 }
220                 if (e.getKeyCode() == KeyEvent.VK_DOWN) {
221                     downPressed = true;
222                 }
223                 if (e.getKeyCode() == KeyEvent.VK_UP) {
224                     upPressed = true;
225                 }
226             }
227
228             public void keyReleased(KeyEvent e) {
229                 if (e.getKeyCode() == KeyEvent.VK_LEFT) {
230                     leftPressed = false;
231                 }
232                 if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
233                     rightPressed = false;
234                 }
235                 if (e.getKeyCode() == KeyEvent.VK_DOWN) {
236                     downPressed = false;
237                 }
238                 if (e.getKeyCode() == KeyEvent.VK_UP) {
239                     upPressed = false;
240                 }
241             }
242         };
243
244         private void isGameOver() {
245             if (hero.getX()==mob.getX() && hero.getY()==mob.getY()) {
246                 running = false;
247             }
248         }
249
```

```
250     private void isWin() {
251         if (hero.getX()==0 && hero.getY()==450) {
252             running = false;
253         }
254     }
255 
256     public void moveForMob() {
257         if (((hero.getX()<mob.getX() && hero.getY()<mob.getY() && (
    ↪ mob.getX()-hero.getX())>(mob.getY()-hero.getY())) || (hero.
    ↪ getX()<mob.getX() && mob.getY()==hero.getY()) || (hero.getX
    ↪ ()<mob.getX() && hero.getY()>mob.getY() && (mob.getX()-hero.
    ↪ getX())>(hero.getY()-mob.getY()))) && playfield.
    ↪ ifPossibleToMove((mob.getX()-50)/50,mob.getY()/50))) {
258             mob.moveX(-50);
259         }
260         if (((hero.getX()>mob.getX() && hero.getY()<mob.getY() && (
    ↪ hero.getX()-mob.getX())>(mob.getY()-hero.getY())) || (hero.
    ↪ getX()>mob.getX() && hero.getY()==mob.getY()) || (hero.getX
    ↪ ()>mob.getX() && hero.getY()>mob.getY() && (hero.getX()-mob.
    ↪ getX())>(hero.getY()-mob.getY())))) && playfield.
    ↪ ifPossibleToMove((mob.getX()+50)/50,mob.getY()/50)) {
261             mob.moveX(50);
262         }
263         if (((hero.getX()<mob.getX() && hero.getY()<mob.getY() && (
    ↪ mob.getX()-hero.getX())<(mob.getY()-hero.getY())) || (hero.
    ↪ getX()==mob.getX() && mob.getY()>hero.getY()) || (hero.getX
    ↪ ()>mob.getX() && hero.getY()<mob.getY() && (hero.getX()-mob.
    ↪ getX())<(mob.getY()-hero.getY())))) && playfield.
    ↪ ifPossibleToMove(mob.getX()/50,(mob.getY()-50)/50)) {
264             mob.moveY(-50);
265         }
266         if (((hero.getX()<mob.getX() && hero.getY()>mob.getY() && (
    ↪ mob.getX()-hero.getX())<(hero.getY()-mob.getY())) || (hero.
    ↪ getX()==mob.getX() && mob.getY()<hero.getY()) || (hero.getX
    ↪ ()>mob.getX() && hero.getY()>mob.getY() && (hero.getX()-mob.
    ↪ getX())<(hero.getY()-mob.getY())))) && playfield.
    ↪ ifPossibleToMove(mob.getX()/50,(mob.getY()+50)/50)) {
267             mob.moveY(50);
268         }
269 
270         if ((hero.getX()<mob.getX() && hero.getY()<mob.getY() && (
    ↪ mob.getX()-hero.getX())==(mob.getY()-hero.getY())) &&
    ↪ playfield.ifPossibleToMove((mob.getX()-50)/50,(mob.getY()
    ↪ -50)/50)) {
271             mob.moveX(-50);
272             mob.moveY(-50);
273         }
274         if ((hero.getX()>mob.getX() && hero.getY()<mob.getY() && (
    ↪ hero.getX()-mob.getX())==(mob.getY()-hero.getY())) &&
    ↪ playfield.ifPossibleToMove((mob.getX()+50)/50,(mob.getY()
    ↪ -50)/50)) {
275             mob.moveX(50);
276             mob.moveY(-50);
277         }
278         if ((hero.getX()<mob.getX() && hero.getY()>mob.getY() && (
    ↪ mob.getX()-hero.getX())==(hero.getY()-mob.getY())) &&
    ↪ playfield.ifPossibleToMove((mob.getX()-50)/50,(mob.getY()
    ↪ +50)/50)) {
279             mob.moveX(-50);
280             mob.moveY(50);
281         }
282         if ((hero.getX()>mob.getX() && hero.getY()>mob.getY() && (
```

```
    ↪ hero.getX()-mob.getX())==(hero.getY()-mob.getY())) &&
    ↪ playfield.ifPossibleToMove((mob.getX()+50)/50,(mob.getY()
    ↪ +50)/50)) {
283             mob.moveX(50);
284             mob.moveY(50);
285         }
286     }
287 }
```

```
 1 package GUI;
 2
 3 import java.awt.event.KeyAdapter;
 4 import java.awt.event.KeyEvent;
 5
 6 /**
 7  * Created by Константин on 05.03.2017.
 8  */
 9 public class KeyInputHandler extends KeyAdapter {
10
11     private boolean leftPressed = false;
12     private boolean rightPressed = false;
13     private boolean downPressed = false;
14     private boolean upPressed = false;
15
16     public void keyPressed(KeyEvent e) {
17         if (e.getKeyCode() == KeyEvent.VK_LEFT) {
18             leftPressed = true;
19         }
20         if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
21             rightPressed = true;
22         }
23         if (e.getKeyCode() == KeyEvent.VK_DOWN) {
24             downPressed = true;
25         }
26         if (e.getKeyCode() == KeyEvent.VK_UP) {
27             upPressed = true;
28         }
29     }
30     public void keyReleased(KeyEvent e) {
31         if (e.getKeyCode() == KeyEvent.VK_LEFT) {
32             leftPressed = false;
33         }
34         if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
35             rightPressed = false;
36         }
37         if (e.getKeyCode() == KeyEvent.VK_DOWN) {
38             downPressed = false;
39         }
40         if (e.getKeyCode() == KeyEvent.VK_UP) {
41             upPressed = false;
42         }
43     }
44 }
```