GUI-Variante des Dijkstra-Algorithmus

Klasse Algorithmus

```
1  public class Algorithmus {
2
3    Knoten[] graph;
4
5    Knoten startknoten = null;
6    Knoten zielknoten = null;
7
8    public Algorithmus() {
9      this.initialisiere();
10   }
11
12   public void initialisiere() {
13     graph = new Knoten[19];
14     graph[1] = new Knoten("A", 257, 18);
15     graph[2] = new Knoten("B", 297, 229);
16     graph[1].fuegeKanteHinzu(new Kante(graph[2], 69));
17     // ...
18   }
19
20   public void berechneWeg() {
21     Warteschlange ws = new Warteschlange();
22     ws.fuegeKnotenEin(this.startknoten);
23
24     Knoten aktuellerKnoten = this.startknoten;
25
26     while (aktuellerKnoten != zielknoten) {
27       aktuellerKnoten.expandiereKnoten(ws);
28       aktuellerKnoten = ws.gibErstesElementZurueck();
29     }
30   }
31
32   public static void main(String[] argv) {
33     Algorithmus algo = new Algorithmus();
34     algo.startknoten = algo.graph[9];
35     algo.zielknoten = algo.graph[14];
36     algo.berechneWeg();
37     algo.zielknoten.gibPfadAus();
38   }
39 }
```

Klasse Kante

```
1  public class Kante {
2
3    public Knoten zielknoten;
4    public int entfernung;
5
6    public Kante(Knoten knoten, int entfernung) {
7      this.zielknoten = knoten;
8      this.entfernung = entfernung;
9    }
10 }
```

Klasse Knoten

```
1  public class Knoten {
2
3    public int entfernung = 0;
4    public Knoten nachfolger;
5    public String name;
6    public Knoten vorgaengerImPfad;
7    public Kante[] kanten = new Kante[10];
8    public int kantenanzahl = 0;
9    public boolean istMarkiert = false;
10   // Neu in der GUI-Variante: Koordinaten des Knotens
11   int x;
12   int y;
13
14   // Neu in der GUI-Variante: Konstruktor mit Koordinaten
15   public Knoten(String name, int koordX, int koordY) {
16     this.name = name;
17     this.x = koordX;
18     this.y = koordY;
19   }
20
21   public void fuegeKnotenEin(Knoten knoten) {
22     if (nachfolger == null) {
23       nachfolger = knoten;
24     } else {
25       if (knoten.entfernung < nachfolger.entfernung) {
26         knoten.nachfolger = nachfolger;
27         nachfolger = knoten;
28       } else {
29         nachfolger.fuegeKnotenEin(knoten);
30       }
31     }
32   }
33
34   public void entferneKnoten(Knoten knoten) {
35     if (nachfolger != null) {
36       if (nachfolger == knoten) {
37         nachfolger = knoten.nachfolger;
38       } else {
39         nachfolger.entferneKnoten(knoten);
40       }
41     }
42   }
43
44   public void fuegeKanteHinzu(Kante kt) {
45     this.kanten[kantenanzahl] = kt;
46     this.kantenanzahl = this.kantenanzahl + 1;
47   }
48
49   public void gibRestWarteschlangeAus() {
50     System.out.println("Knoten: " + name);
51     if (nachfolger != null) {
52       nachfolger.gibRestWarteschlangeAus();
53     }
54   }
55
56   public void gibPfadAus() {
57     if (this.vorgaengerImPfad != null) {
58       this.vorgaengerImPfad.gibPfadAus();
59     }
60     System.out.println("Knoten: " + name);
61   }
```

```
62
63    // Neu in der GUI-Variante: Abstand des Knotens zu einem Punkt
64    public float berechneMausDistanz(int x1, int y1) {
65      return (float) Math.sqrt(Math.pow(x - x1, 2) + Math.pow(y - y1, 2));
66    }
67
68    public void expandiereKnoten(Warteschlange ws) {
69      for (int i = 0; i < this.kantenanzahl; i++) {
70        Knoten nachbarknoten = kanten[i].zielknoten;
71        int kantenentfernung = kanten[i].entfernung;
72        if (!nachbarknoten.istMarkiert) {
73          if (nachbarknoten.entfernung == 0) {
74            nachbarknoten.entfernung = kantenentfernung
75                + this.entfernung;
76            nachbarknoten.vorgaengerImPfad = this;
77            ws.fuegeKnotenEin(nachbarknoten);
78          } else {
79            if (nachbarknoten.entfernung > kantenentfernung
80                + this.entfernung) {
81              ws.entferneKnoten(nachbarknoten);
82              nachbarknoten.entfernung = kantenentfernung
83                  + this.entfernung;
84              nachbarknoten.vorgaengerImPfad = this;
85              ws.fuegeKnotenEin(nachbarknoten);
86            }
87          }
88        }
89        this.istMarkiert = true;
90      }
91    }
92  }
```

Klasse Warteschlange

```
1  public class Warteschlange {
2
3    private Knoten kopf;
4
5    public Warteschlange() {
6      kopf = new Knoten("Kopf", -1);
7    }
8
9    public void fuegeKnotenEin(Knoten knoten) {
10     kopf.fuegeKnotenEin(knoten);
11   }
12
13   public void entferneKnoten(Knoten knoten) {
14     kopf.entferneKnoten(knoten);
15   }
16
17   public Knoten gibErstesElementZurueck() {
18     Knoten hilfsknoten = kopf.nachfolger;
19     kopf.entferneKnoten(kopf.nachfolger);
20     return hilfsknoten;
21
22   }
23
24
25   public void gibWarteschlangeAus() {
26     kopf.gibRestWarteschlangeAus();
27   }
28 }
```

Klasse GUIeinfach

```
1  import java.awt.BasicStroke;
2  import java.awt.Color;
3  import java.awt.Font;
4  import java.awt.Graphics2D;
5  import java.awt.Image;
6  import java.awt.MediaTracker;
7  import java.awt.Toolkit;
8  import java.awt.event.MouseEvent;
9  import java.awt.event.MouseListener;
10 import javax.swing.JFrame;
11
12 public class GUIeinfach extends JFrame {
13
14   Algorithmus algorithmus;
15   Graphics2D graphics;
16
17   public GUIeinfach() {
18     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19     this.setTitle("Dijkstra-Algorithmus");
20     this.initialisiere();
21   }
22
23   public void initialisiere() {
24     algorithmus = new Algorithmus();
25     this.setVisible(true);
26     this.setSize(634, 769);
27     this.graphics = (Graphics2D) this.getContentPane().getGraphics();
28     Image img = Toolkit.getDefaultToolkit().getImage("Lageplan.gif");
29     MediaTracker mediaTracker = new MediaTracker(this);
30     mediaTracker.addImage(img, 0);
31     try {
32       mediaTracker.waitForID(0);
33     } catch (InterruptedException ie) {
34       System.err.println(ie);
35       System.exit(1);
36     }
37     graphics.drawImage(img, 0, 0, null);
38   }
39
40   public void zeichneKnoten(Knoten k, Color farbe) {
41     graphics.setColor(farbe);
42     graphics.fillOval(k.x - 18, k.y - 18, 37, 37);
43     graphics.setColor(Color.black);
44     graphics.setStroke(new BasicStroke(2));
45     graphics.drawOval(k.x - 18, k.y - 18, 37, 37);
46     graphics.setFont(new Font("Helvetica", Font.PLAIN, 24));
47     graphics.drawString(k.name, k.x - 7, k.y + 9);
48   }
49
50   public void zeichneWeg(Knoten knoten, Color farbe) {
51     zeichneKnoten(knoten, farbe);
52     if (knoten.vorgaengerImPfad != null) {
53       zeichneWeg(knoten.vorgaengerImPfad, farbe);
54     }
55   }
56
57   public static void main(String[] argv) {
58     GUIeinfach ui = new GUIeinfach();
59     ui.initialisiere();
60     Algorithmus algo = ui.algorithmus;
61     algo.startknoten = algo.graph[9];
```

```
62      algo.zielknoten = algo.graph[14];
63      algo.berechneWeg();
64      ui.zeichneWeg(algo.zielknoten, Color.blue);
65    }
66  }
```

Klasse GUI

```
1   import java.awt.BasicStroke;
2   import java.awt.Color;
3   import java.awt.Font;
4   import java.awt.Graphics2D;
5   import java.awt.Image;
6   import java.awt.MediaTracker;
7   import java.awt.Toolkit;
8   import java.awt.event.MouseEvent;
9   import java.awt.event.MouseListener;
10  import javax.swing.JFrame;
11
12  public class GUI extends JFrame implements MouseListener {
13
14    Algorithmus algorithmus;
15    Graphics2D graphics;
16
17    public GUI() {
18      this.addMouseListener(this);
19      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20      this.setTitle("Dijkstra-Algorithmus");
21      this.initialisiere();
22    }
23
24    public void initialisiere() {
25      algorithmus = new Algorithmus();
26      this.setVisible(true);
27      this.setSize(634, 769);
28      this.graphics = (Graphics2D) this.getContentPane().getGraphics();
29      Image img = Toolkit.getDefaultToolkit().getImage("Lageplan.gif");
30      MediaTracker mediaTracker = new MediaTracker(this);
31      mediaTracker.addImage(img, 0);
32      try {
33        mediaTracker.waitForID(0);
34      } catch (InterruptedException ie) {
35        System.err.println(ie);
36        System.exit(1);
37      }
38      graphics.drawImage(img, 0, 0, null);
39    }
40
41    /**
42     * findet den Knoten, der am naechsten an dem angegebenen Punkt liegt.
43     */
44    public Knoten findeNaechstenKnoten(int mouseX, int mouseY) {
45      Knoten knoten[] = algorithmus.graph;
46      int zaehler = 1;
47      Knoten aktuellerKnoten = knoten[zaehler];
48      Knoten naechsterKnoten = aktuellerKnoten;
49      int aktuelleDistanz =
50              (int) naechsterKnoten.berechneMausDistanz(mouseX, mouseY);
51      zaehler++;
52      while (zaehler < 19) {
53        aktuellerKnoten = knoten[zaehler];
```

```
54          if (aktuellerKnoten.berechneMausDistanz(mouseX, mouseY) < aktuelleDistanz) {
55            naechsterKnoten = aktuellerKnoten;
56            aktuelleDistanz =
57                (int) aktuellerKnoten.berechneMausDistanz(mouseX, mouseY);
58          }
59          zaehler++;
60        }
61        return naechsterKnoten;
62      }
63
64      public void zeichneKnoten(Knoten k, Color farbe) {
65        graphics.setColor(farbe);
66        graphics.fillOval(k.x - 18, k.y - 18, 37, 37);
67        graphics.setColor(Color.black);
68        graphics.setStroke(new BasicStroke(2));
69        graphics.drawOval(k.x - 18, k.y - 18, 37, 37);
70        graphics.setFont(new Font("Helvetica", Font.PLAIN, 24));
71        graphics.drawString(k.name, k.x - 7, k.y + 9);
72      }
73
74      public void zeichneWeg(Knoten knoten, Color farbe) {
75        zeichneKnoten(knoten, farbe);
76        if (knoten.vorgaengerImPfad != null) {
77          zeichneWeg(knoten.vorgaengerImPfad, farbe);
78        }
79      }
80
81      // Methoden des Mouse-Listener
82      public void mouseClicked(MouseEvent e) {
83        if (algorithmus.startknoten == null &&
84                  algorithmus.zielknoten == null) {
85          algorithmus.startknoten = this.findeNaechstenKnoten(e.getX(), e.getY());
86          this.zeichneKnoten(algorithmus.startknoten, Color.red);
87        } else if (algorithmus.startknoten != null &&
88                  algorithmus.zielknoten == null) {
89          algorithmus.zielknoten = this.findeNaechstenKnoten(e.getX(), e.getY());
90          this.zeichneKnoten(algorithmus.zielknoten, Color.green);
91          algorithmus.berechneWeg();
92          this.zeichneWeg(algorithmus.zielknoten, Color.blue);
93        } else {
94          algorithmus.zielknoten = null;
95          algorithmus.startknoten = null;
96          this.initialisiere();
97        }
98      }
99
100     public void mouseEntered(MouseEvent e) {
101     }
102
103     public void mouseExited(MouseEvent e) {
104     }
105
106     public void mousePressed(MouseEvent e) {
107     }
108
109     public void mouseReleased(MouseEvent e) {
110     }
111
112     public static void main(String[] argv) {
113       GUI ui = new GUI();
114       ui.initialisiere();
115     }
116 }
```