

Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung:

Lösungsvorschlag

Aufgabe 8-1

AWT/Swing

Präsenz

In dieser Aufgabe sollen Sie eine grafische Benutzeroberfläche implementieren, mit der eine Figur erzeugt, bewegt und gelöscht werden kann. Nehmen Sie für diese Aufgabe die Klasse **Figur** von Übungsblatt 6, Aufgabe 6-3, zur Grundlage.

Hinweis: Damit Ihr Programm korrekt auf die Klasse **Figur** und die Klasse **Point** (die von der Klasse **Figur** benötigt wird) zugreifen kann, entpacken Sie die beiden Klassen aus dem ZIP-Archiv und speichern Sie im gleichen Ordner wie Ihre neuen Klassen **FigurFrame** und **FigurFrameMain**.

a) Die grafische Benutzeroberfläche soll wie folgt aussehen:



Es soll drei Buttons zum Erzeugen, Löschen und Bewegen einer Figur geben, die in der ersten Zeile direkt nebeneinander angeordnet sind. In der zweiten Zeile soll ein Ausgabebereich platziert werden, in dem Rückmeldung über das Erzeugen, Löschen oder Bewegen einer Figur gegeben werden kann.

Schreiben Sie eine Klasse **FigurFrame**, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse **FigurFrameMain**, die Sie im gleichen Ordner wie Ihre Klasse **FigurFrame** abspeichern. Die Klasse **FigurFrameMain** soll ein Objekt vom Typ **FigurFrame** erzeugen und das Fenster sichtbar machen.

Lösung:

Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse **FigurFrame**, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse **JFrame** erbt. Ihre Klasse **FigurFrame** soll vier Attribute haben:

- je ein Attribut vom Klassentyp **JButton** für Buttons zum Erzeugen, Löschen und Bewegen einer Figur,
- ein Attribut vom Klassentyp **JTextArea**, das als Ausgabebereich für die spätere Rückmeldung über die Erzeugung, Löschung und Bewegung der Figur dient,

Schreiben Sie einen Konstruktor, der den **FigurFrame** mit einem entsprechenden Titel und Größe (wir wählen hier 700x200 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Alle Buttons sollen in einem **JPanel** gruppiert werden. Das Layout des **ContentPane** des **FigurFrames** wird auf ein **GridLayout** mit zwei Zeilen und einer Spalte initialisiert und das Button-JPanel sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der **FigurFrame** geschlossen wird. Benutzen Sie dazu die Methode **setDefaultCloseOperation**.

Weiter unten finden Sie eine Implementierung der Klassen **FigurFrame und **FigurFrameMain** für die gesamte Aufgabe.**

- b) Deklarieren Sie in Ihrer Klasse **FigurFrame** ein Attribut **aktuelleFigur** mit Klassentyp **Figur**. Erweitern Sie Ihre Klasse **FigurFrame** um eine Ereignisbehandlung für den Button zur Erzeugung einer Figur. Wird der Button zur Erzeugung gedrückt, soll eine neue Figur erzeugt und im Attribut **aktuelleFigur** gespeichert werden. Ist in dem Attribut bereits eine Figur gespeichert, soll keine neue Figur erzeugt werden, sondern eine Fehlermeldung im Ausgabebereich angezeigt werden. Alle nötigen Informationen zur Erzeugung einer Figur sollen vom Benutzer mit Hilfe der Klasse **JOptionPane** abgefragt werden. Nach der Erzeugung einer Figur soll der Benutzer im Ausgabebereich darüber informiert werden, d.h. es soll ausgegeben werden, welches Objekt erzeugt wurde und welche Koordinaten dessen Mittelpunkt hat.

Lösung:

Bei der Implementierung dieser Aufgabe müssen Sie bedenken, dass für den Button zunächst der **FigurFrame** als **ActionListener** registriert werden muss. Der **FigurFrame** muss dann eine Methode **actionPerformed** implementieren, die überprüft, welcher Button gedrückt wurde. Falls der Button zur Erzeugung gedrückt wurde, wird eine neue Methode aufgerufen, die die Erzeugung einer Figur, wie oben beschrieben, vornimmt.

Weiter unten finden Sie eine Implementierung der Klassen **FigurFrame und **FigurFrameMain** für die gesamte Aufgabe.**

- c) Erweitern Sie Ihre Klasse **FigurFrame** um eine Ereignisbehandlung für den Button zum Löschen einer Figur. Wird der Button zum Löschen gedrückt, soll die aktuell gespeicherte Figur gelöscht werden. Dazu müssen Sie lediglich dem Attribut **aktuelleFigur** für die aktuelle Figur den Wert **null** zuweisen. Nach dem Löschen der Figur soll der Benutzer im Ausgabebereich darüber informiert werden. Falls allerdings keine aktuelle Figur gespeichert war, soll nach dem Knopfdruck eine Fehlermeldung im Ausgabebereich angezeigt werden.

Lösung:

Bei der Implementierung dieser Aufgabe müssen Sie bedenken, dass für diesen Button auch der **FigurFrame** als **ActionListener** registriert werden muss und die Methode **actionPerformed** erweitert werden muss.

Weiter unten finden Sie eine Implementierung der Klassen **FigurFrame und **FigurFrameMain** für die gesamte Aufgabe.**

- d) Erweitern Sie Ihre Klasse **FigurFrame** um eine Ereignisbehandlung für den Button zum Bewegen einer Figur. Wird der Button zum Bewegen einer Figur gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse **JOptionPane** gefragt werden, um wieviel die Figur bewegt werden soll (d.h. um wieviel die x-Koordinate des Mittelpunkts und um wieviel die y-Koordinate des Mittelpunkts bewegt werden soll), und anschließend die gespeicherte Figur entsprechend bewegt werden. Nach dem Bewegen soll der Benutzer im Ausgabebereich darüber informiert werden, d.h. es soll ausgegeben werden, welche Koordinaten der Mittelpunkt der Figur nun

hat. Falls keine aktuelle Figur gespeichert ist, soll eine Fehlermeldung im Ausgabebereich angezeigt werden.

Lösung:

Bei der Implementierung dieser Aufgabe müssen Sie bedenken, dass für diesen Button auch der FigurFrame als ActionListener registriert werden muss und die Methode actionPerformed erweitert werden muss.

Die Implementierung der Klassen FigurFrame und FigurFrameMain finden Sie auch im ZIP-Archiv.

```
1 import java.awt.Container;
2 import java.awt.GridLayout;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 import javax.swing.JButton;
7 import javax.swing.JFrame;
8 import javax.swing.JOptionPane;
9 import javax.swing.JPanel;
10 import javax.swing.JTextArea;
11
12 public class FigurFrame extends JFrame implements ActionListener {
13
14     private JButton erzeugenButton;
15     private JButton loeschenButton;
16     private JButton bewegenButton;
17
18     private JTextArea ausgabeBereich;
19
20     private Figur aktuelleFigur;
21
22     /**
23      * In diesem Programmstueck wird das Fenster erzeugt.
24      * Auf dem Fenster sind spaeter drei Buttons fuer Erzeugen,
25      * Loeschen und Bewegen zu sehen. Ausserdem gibt es einen
26      * Ausgabebereich, in dem ausgegeben wird, welche Aktion das
27      * Programm gerade ausgefuehrt hat.
28      */
29     public FigurFrame() {
30         /* In der Kopfleiste des Fenster steht "FigurFrame" */
31         this.setTitle("FigurFrame");
32
33         /* Das Fenster ist 700 Pixel breit und 200 Pixel hoch. */
34         this.setSize(700, 200);
35
36         /* Hier werden alle Buttons erzeugt. */
37         this.erzeugenButton = new JButton("Erzeugen");
38         this.loeschenButton = new JButton("Loeschen");
39         this.bewegenButton = new JButton("Bewegen");
40
41         /* Hier wird der Ausgabe-Bereich erzeugt. */
42         this.ausgabeBereich = new JTextArea(10, 100);
43
44         /* Hier werden alle Buttons zusammengruppiert. */
45         JPanel buttonPanel = new JPanel();
46         buttonPanel.add(this.erzeugenButton);
47         buttonPanel.add(this.loeschenButton);
48         buttonPanel.add(this.bewegenButton);
49
50         /*
51          * Der ContentPane ist der Ausschnitt des Fensters,
52          * auf dem Widgets d.h. Interaktionselemente
53          * (wie eine TextArea oder ein Button) platziert werden koennen.
54          */
55     }
```

```

55     Container contentPane = this.getContentPane();
56     contentPane.setLayout(new GridLayout(2, 1));
57     /* Hier wird die Gruppe von Buttons platziert. */
58     contentPane.add(buttonPanel);
59     /* Hier wird der Ausgabebereich platziert. */
60     contentPane.add(this.ausgabeBereich);
61
62     /*
63      * Hier wird der FigurFrame als Listener fuer Knopfdruck-Ereignisse
64      * bei jedem der drei Buttons registriert.
65      */
66     this.erzeugenButton.addActionListener(this);
67     this.loeschenButton.addActionListener(this);
68     this.bewegenButton.addActionListener(this);
69
70     /*
71      * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
72      * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
73      */
74     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
75 }
76
77 /**
78  * Dieses Programmstueck wird immer dann ausgefuehrt,
79  * wenn ein Benutzer auf einen Button drueckt.
80  */
81 @Override
82 public void actionPerformed(ActionEvent e) {
83     Object source = e.getSource();
84     if (source == this.erzeugenButton) {
85         this.erzeugen();
86     }
87     else if (source == this.loeschenButton) {
88         this.loeschen();
89     }
90     else if (source == this.bewegenButton) {
91         this.bewegen();
92     }
93 }
94
95 /**
96  * Diese Methode erzeugt eine neue Figur, falls es momentan keine Figur
97  * gibt.
98  */
99 private void erzeugen() {
100     if (this.aktuelleFigur != null) {
101         this.ausgabeBereich
102             .setText("Es existiert bereits eine Figur. "
103                 + "Sie muessen diese Figur zuerst loeschen, "
104                 + "bevor eine neue Figur erzeugt werden kann!");
105     }
106     else {
107         String einlesenPositionX = JOptionPane
108             .showInputDialog("Mittelpunkt x-Position: ");
109         int xpos = Integer.parseInt(einlesenPositionX);
110
111         String einlesenPositionY = JOptionPane
112             .showInputDialog("Mittelpunkt y-Position: ");
113         int ypos = Integer.parseInt(einlesenPositionY);
114
115         String farbe = JOptionPane.showInputDialog("Farbe: ");
116
117         String einlesenFuellung = JOptionPane
118             .showInputDialog("Fuellung [true/false]: ");
119         boolean fuellung = Boolean.parseBoolean(einlesenFuellung);

```

```

120
121         this.aktuelleFigur = new Figur(xpos, ypos, farbe,
122             fuellung);
123
124         this.ausgabeBereich
125             .setText("Folgendes Objekt wurde erzeugt: "
126                 + this.aktuelleFigur
127                 + " mit dem Mittelpunkt: "
128                 + this.aktuelleFigur.getMittelpunkt()
129                     .getX()
130                 + ", "
131                 + this.aktuelleFigur.getMittelpunkt()
132                     .getY());
133     }
134 }
135
136 /**
137  * Diese Methode loescht die aktuelle Figur, falls es eine gibt.
138  */
139 private void loeschen() {
140     if (this.aktuelleFigur == null) {
141         this.ausgabeBereich.setText("Es existiert keine Figur, "
142             + "die geloescht werden kann.");
143     }
144     else {
145         this.aktuelleFigur = null;
146         this.ausgabeBereich.setText("Die Figur wurde geloescht.");
147     }
148 }
149
150 /**
151  * Diese Methode bewegt die aktuelle Figur, falls es eine gibt.
152  */
153 private void bewegen() {
154     if (this.aktuelleFigur == null) {
155         this.ausgabeBereich.setText("Es existiert keine Figur, "
156             + "die bewegt werden kann.");
157     }
158     else {
159         String einlesenDx = JOptionPane
160             .showInputDialog("Objekt bewegen um x-Wert: ");
161         int dx = Integer.parseInt(einlesenDx);
162
163         String einlesenDy = JOptionPane
164             .showInputDialog("Objekt bewegen um y-Wert: ");
165         int dy = Integer.parseInt(einlesenDy);
166
167         this.aktuelleFigur.bewegen(dx, dy);
168
169         this.ausgabeBereich.setText("Das Objekt "
170             + this.aktuelleFigur
171             + " hat nun den Mittelpunkt: "
172             + this.aktuelleFigur.getMittelpunkt().getX()
173             + ", "
174             + this.aktuelleFigur.getMittelpunkt().getY());
175     }
176 }
177 }
178 }

```

```

1 /**
2  * Diese Klasse startet den FigurFrame
3  *
4  * @author Annabelle Klarl
5  *

```

```

6  */
7  public class FigurFrameMain {
8
9      /**
10       * Dieses Programmstueck startet das Programm.
11       *
12       * @param args
13       */
14     public static void main(String[] args) {
15         FigurFrame figurFrame = new FigurFrame();
16         figurFrame.setVisible(true);
17     }
18
19 }

```

Aufgabe 8-2

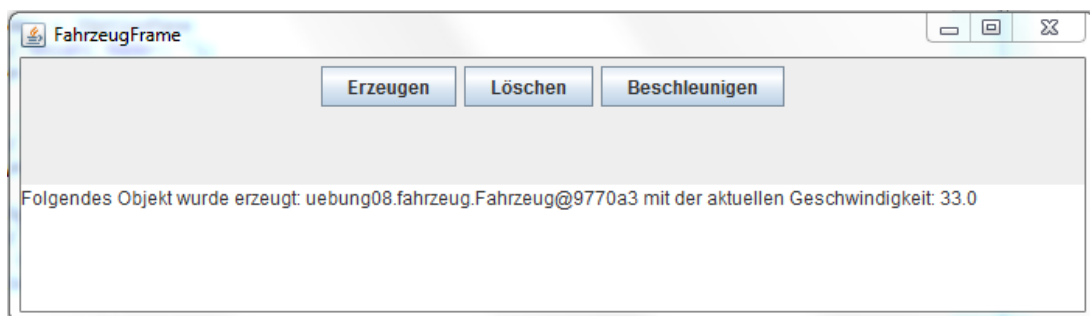
AWT/Swing

Hausaufgabe

In dieser Aufgabe sollen Sie eine grafische Benutzeroberfläche implementieren, mit der ein Fahrzeug erzeugt, beschleunigt und gelöscht werden kann. Nehmen Sie für diese Aufgabe die Klasse **Fahrzeug** von Übungsblatt 6, Aufgabe 6-4, zur Grundlage.

Hinweis: Damit Ihr Programm korrekt auf die Klasse **Fahrzeug** und die Klasse **Point** (die von der Klasse **Fahrzeug** benötigt wird) zugreifen kann, entpacken Sie die beiden Klassen aus dem ZIP-Archiv und speichern Sie im gleichen Ordner wie Ihre neuen Klassen **FahrzeugFrame** und **FahrzeugFrameMain**.

a) Die grafische Benutzeroberfläche soll wie folgt aussehen:



Es soll drei Buttons zum Erzeugen, Löschen und Beschleunigen eines Fahrzeugs geben, die in der ersten Zeile direkt nebeneinander angeordnet sind. In der zweiten Zeile soll ein Ausgabebereich platziert werden, in dem Rückmeldung über das Erzeugen, Löschen oder Beschleunigen eines Fahrzeugs gegeben werden kann.

Schreiben Sie eine Klasse **FahrzeugFrame**, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse **FahrzeugFrameMain**, die Sie im gleichen Ordner wie Ihre Klasse **FahrzeugFrame** abspeichern. Die Klasse **FahrzeugFrameMain** soll ein Objekt vom Typ **FahrzeugFrame** erzeugen und das Fenster sichtbar machen.

Lösung:

Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse **FahrzeugFrame**, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse **JFrame** erbt. Ihre Klasse **FahrzeugFrame** soll vier Attribute haben:

- je ein Attribut vom Klassentyp `JButton` für Buttons zum Erzeugen, Löschen und Beschleunigen eines Fahrzeugs,
- ein Attribut vom Klassentyp `JTextArea`, das als Ausgabebereich für die spätere Rückmeldung über die Erzeugung, Löschung und Beschleunigung des Fahrzeugs dient,

Schreiben Sie einen Konstruktor, der den `FahrzeugFrame` mit einem entsprechenden Titel und Größe (wir wählen hier 700x200 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Alle Buttons sollen in einem `JPanel` gruppiert werden. Das Layout des `ContentPane` des `FahrzeugFrames` wird auf ein `GridLayout` mit zwei Zeilen und einer Spalte initialisiert und das Button-`JPanel` sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der `FahrzeugFrame` geschlossen wird. Benutzen Sie dazu die Methode `setDefaultCloseOperation`.

Weiter unten finden Sie eine Implementierung der Klassen `FahrzeugFrame` und `FahrzeugFrameMain` für die gesamte Aufgabe.

- b) Erweitern Sie Ihre Klasse `FahrzeugFrame` um Ereignisbehandlungen für die Buttons zur Erzeugung, zum Löschen und Beschleunigen eines Fahrzeug.

Lösung:

Die Implementierung der Klassen `FahrzeugFrame` und `FahrzeugFrameMain` finden Sie auch im ZIP-Archiv.

```

1 import java.awt.Container;
2 import java.awt.GridLayout;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 import javax.swing.JButton;
7 import javax.swing.JFrame;
8 import javax.swing.JOptionPane;
9 import javax.swing.JPanel;
10 import javax.swing.JTextArea;
11
12 public class FahrzeugFrame extends JFrame implements ActionListener {
13
14     private JButton erzeugenButton;
15     private JButton loeschenButton;
16     private JButton beschleunigenButton;
17
18     private JTextArea ausgabeBereich;
19
20     private Fahrzeug aktuellesFahrzeug;
21
22     /**
23      * In diesem Programmstueck wird das Fenster erzeugt. Auf dem Fenster sind
24      * spaeter drei Buttons fuer Erzeugen, Loeschen und Beschleunigen zu sehen.
25      * Ausserdem gibt es einen Ausgabebereich, in dem ausgegeben wird, welche
26      * Aktion das Programm gerade ausgefuehrt hat.
27      */
28     public FahrzeugFrame() {
29         /* In der Kopfleiste des Fenster steht "FahrzeugFrame" */
30         this.setTitle("FahrzeugFrame");
31
32         /* Das Fenster ist 700 Pixel breit und 200 Pixel hoch. */
33         this.setSize(700, 200);
34
35         /* Hier werden alle Buttons erzeugt. */
36         this.erzeugenButton = new JButton("Erzeugen");

```

```

37     this.loeschenButton = new JButton("Loeschen");
38     this.beschleunigenButton = new JButton("Beschleunigen");
39
40     /* Hier wird der Ausgabe-Bereich erzeugt. */
41     this.ausgabeBereich = new JTextArea(10, 100);
42
43     /* Hier werden alle Buttons zusammengruppiert. */
44     JPanel buttonPanel = new JPanel();
45     buttonPanel.add(this.erzeugenButton);
46     buttonPanel.add(this.loeschenButton);
47     buttonPanel.add(this.beschleunigenButton);
48
49     /*
50      * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
51      * Interaktionselemente (wie eine TextArea oder ein Button) platziert
52      * werden koennen.
53     */
54     Container contentPane = this.getContentPane();
55     contentPane.setLayout(new GridLayout(2, 1));
56     /* Hier wird die Gruppe von Buttons platziert. */
57     contentPane.add(buttonPanel);
58     /* Hier wird der Ausgabebereich platziert. */
59     contentPane.add(this.ausgabeBereich);
60
61     /*
62      * Hier wird der FahrzeugFrame als Listener fuer Knopfdruck-Ereignisse
63      * bei jedem der drei Buttons registriert.
64     */
65     this.erzeugenButton.addActionListener(this);
66     this.loeschenButton.addActionListener(this);
67     this.beschleunigenButton.addActionListener(this);
68
69     /*
70      * Wird das Fenster geschlossen (d.h. auf X gedruickt), wird mit Hilfe
71      * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
72     */
73     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
74 }
75
76 /**
77  * Dieses Programmstueck wird immer dann ausgefuehrt, wenn ein Benutzer auf
78  * einen Button drueckt.
79 */
80 @Override
81 public void actionPerformed(ActionEvent e) {
82     Object source = e.getSource();
83     if (source == this.erzeugenButton) {
84         this.erzeugen();
85     }
86     else if (source == this.loeschenButton) {
87         this.loeschen();
88     }
89     else if (source == this.beschleunigenButton) {
90         this.beschleunigen();
91     }
92 }
93
94 /**
95  * Diese Methode erzeugt ein neues Fahrzeug, falls es momentan kein Fahrzeug
96  * gibt.
97 */
98 private void erzeugen() {
99     if (this.aktuellesFahrzeug != null) {
100         this.ausgabeBereich
101             .setText("Es existiert bereits ein Fahrzeug. "

```



```

102         + "Sie muessen dieses Fahrzeug zuerst loeschen, "
103         + "bevor ein neues Fahrzeug erzeugt werden kann!"));
104     }
105     else {
106         String einlesenPositionX = JOptionPane
107             .showInputDialog("Aktuelle x-Position: ");
108         int xpos = Integer.parseInt(einlesenPositionX);
109
110         String einlesenPositionY = JOptionPane
111             .showInputDialog("Aktuelle y-Position: ");
112         int ypos = Integer.parseInt(einlesenPositionY);
113
114         String einlesenAnzahlRaeder = JOptionPane
115             .showInputDialog("Anzahl Raeder: ");
116         int anzahlRaeder = Integer.parseInt(einlesenAnzahlRaeder);
117
118         String einlesenGewicht = JOptionPane
119             .showInputDialog("Gewicht: ");
120         double gewicht = Double.parseDouble(einlesenGewicht);
121
122         String einlesenGeschwindigkeit = JOptionPane
123             .showInputDialog("Geschwindigkeit: ");
124         double geschwindigkeit = Double
125             .parseDouble(einlesenGeschwindigkeit);
126
127         this.aktuellesFahrzeug = new Fahrzeug(xpos, ypos,
128             anzahlRaeder, gewicht, geschwindigkeit);
129
130         this.ausgabeBereich
131             .setText("Folgendes Objekt wurde erzeugt: "
132                 + this.aktuellesFahrzeug
133                 + " mit der aktuellen Geschwindigkeit: "
134                 + this.aktuellesFahrzeug
135                 .getAktuelleGeschwindigkeit());
136     }
137 }
138
139 /**
140  * Diese Methode loescht das aktuelle Fahrzeug, falls es eines gibt.
141  */
142 private void loeschen() {
143     if (this.aktuellesFahrzeug == null) {
144         this.ausgabeBereich
145             .setText("Es existiert kein Fahrzeug, "
146                 + "das geloescht werden kann.");
147     }
148     else {
149         this.aktuellesFahrzeug = null;
150         this.ausgabeBereich
151             .setText("Das Fahrzeug wurde geloescht.");
152     }
153 }
154
155 /**
156  * Diese Methode beschleunigt das aktuelle Fahrzeug, falls es eines gibt.
157  */
158 private void beschleunigen() {
159     if (this.aktuellesFahrzeug == null) {
160         this.ausgabeBereich
161             .setText("Es existiert kein Fahrzeug, "
162                 + "das beschleunigt werden kann.");
163     }
164     else {
165         String einlesenBeschleunigung = JOptionPane

```

```

167         .showInputDialog("Um wieviel moechten Sie beschleunigen: ");
168         double beschleunigung = Double
169             .parseDouble(einlesenBeschleunigung);
170
171         this.aktuellesFahrzeug.beschleunigen(beschleunigung);
172
173         this.ausgabeBereich.setText("Das Objekt: "
174             + this.aktuellesFahrzeug
175             + " hat nun die aktuelle Geschwindigkeit: "
176             + this.aktuellesFahrzeug
177                 .getAktuelleGeschwindigkeit());
178     }
179 }
180 }

```

```

1  /**
2   * Diese Klasse startet den FahrzeugFrame
3   *
4   * @author Annabelle Klarl
5   *
6   */
7  public class FahrzeugFrameMain {
8
9      /**
10       * Dieses Programmstueck startet das Programm.
11       *
12       * @param args
13       */
14     public static void main(String[] args) {
15         FahrzeugFrame fahrzeugFrame = new FahrzeugFrame();
16         fahrzeugFrame.setVisible(true);
17     }
18
19 }

```

Besprechung der Präsenzaufgaben in den Übungen ab 07.12.2018. Abgabe der Hausaufgaben bis Mittwoch, 19.12.2018, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung).

- Erstellen Sie zu jeder Aufgabe Klassen mit den entsprechenden Namen, die in der Aufgabe gefordert sind.
- Geben Sie nur die entsprechenden *.java*-Dateien ab. Wir benötigen **nicht** Ihre *.class*-Dateien.
- Geben Sie Java-Code nur in *.java*-Dateien ab. Java-Code in Bildern, PDF-Dokumenten und Text-Dateien wird nicht korrigiert.