Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. R. Hennicker, P. Wendler

WS 2017/18
Übungsblatt 13

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

Aufgabe 13-1 **Ballonwettbewerb** Präsenz

In einer Stadt werden mehrere tausend Ballons mit Absenderpostkarten losgelassen. Bis zu einem gewissen Stichtag kommt ein Teil der Postkarten mit Angabe des Fundortes zurück. Sieger ist, wessen Ballon am weitesten geflogen ist (in km), wobei bei gleicher Entfernung mehrere Sieger möglich sind.

a) Schreiben Sie eine Klasse `Postkarte`, die eine Absenderpostkarte wie oben beschrieben repräsentiert. Die Klasse soll Attribute für den Namen des Teilnehmers und die zurückgelegte Entfernung (als Ganzzahl vom Typ `int`) haben. Fügen Sie einen Konstruktor zur Initialisierung der Attribute sowie "Getter" für beide Attribute hinzu.

Lösung:
Eine Implementierung der Klasse `Postkarte` finden Sie auch im ZIP-Archiv.

```java
/**
 * Diese Klasse repraesentiert eine Postkarte bestehend aus dem Teilnehmer,
 * der diese Postkarte anfaenglich losgeschickt hat, und der Entfernung in km
 * (wobei hier nur ganzzahlige Entfernungen erlaubt sind),
 * die diese Postkarte zurueckgelegt hat.
 *
 * @author Annabelle Klarl
 *
 */
public class Postkarte {

    private String teilnehmer;
    private int entfernung;

    /**
     * Konstruktor
     *
     * @param teilnehmer
     * @param entfernung zurueckgelegte Entfernung in km als Ganzzahl
     */
    public Postkarte(String teilnehmer, int entfernung) {
        this.teilnehmer = teilnehmer;
        this.entfernung = entfernung;
    }

    /**
     * Diese Methode gibt den Namen des Teilnehmers zurueck,
     * der diese Postkarte anfaenglich abgeschickt hat.
     *
     * @return der Name des Teilnehmers
     */
    public String getTeilnehmer() {
        return this.teilnehmer;
    }

    /**
     * Diese Methode gibt die Entfernung zurueck,
     * die die Postkarte zurueckgelegt hat.
```

```
39      *
40      * @return die zurueckgelegte Entfernung
41      */
42     public int getEntfernung() {
43         return this.entfernung;
44     }
45
46 }
```

b) Implementieren Sie nun eine verkettete Liste zur Speicherung von Postkarten nach dem Beispiel in der Vorlesung. Nennen Sie die Klasse, die Ihre verkettete Liste darstellt, `MeinePostkartenListe` und die Klasse für Elemente in dieser Liste `MeinPostkartenListenElement`.

Lösung:
Eine Implementierung der Klassen `MeinePostkartenListe` und `MeinPostkartenListenElement` finden Sie auch im ZIP-Archiv.

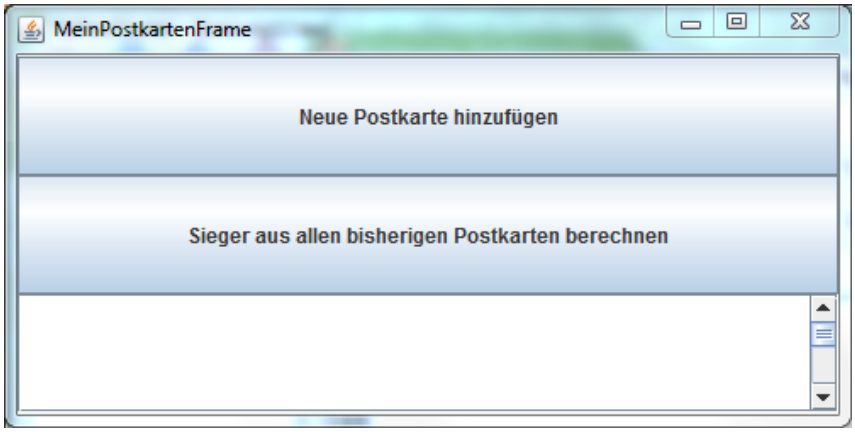
```
1  /**
2   * Diese Klasse implementiert eine verkettete Liste, die Postkarten speichert.
3   * Dazu werden als Elemente Objekte vom Typ {@link MeinPostkartenListenElement}
4   * benutzt.
5   *
6   * @author Annabelle Klarl
7   *
8   */
9  public class MeinePostkartenListe {
10
11     private MeinPostkartenListenElement first;
12
13     /**
14      * Konstruktor: erstellt eine leere Liste
15      */
16     public MeinePostkartenListe() {
17         this.first = null;
18     }
19
20     /**
21      * Fuegt die gegebene Postkarte an erster Stelle in die Liste ein. Alle
22      * bisherigen Listen-Elemente werden um eins nach hinten gerueckt.
23      *
24      * @param postkarte
25      */
26     public void addFirst(Postkarte postkarte) {
27         this.first = new MeinPostkartenListenElement(postkarte, this.first);
28     }
29
30     /**
31      * Bestimmt die Anzahl der Elemente in der Liste
32      *
33      * @return die Anzahl der Elemente in der Liste
34      */
35     public int size() {
36         int size = 0;
37         MeinPostkartenListenElement element = this.first;
38         while (element != null) {
39             element = element.getNext();
40             size++;
41         }
42         return size;
43     }
44
```

```
45      /**
46       * Gibt das Element an Position i in der Liste zurueck
47       * (ohne es aus der Liste zu loeschen!)
48       *
49       * @param i
50       * @return das Listen-Element an Position i
51       */
52      public Postkarte get(int i) {
53          MeinPostkartenListenElement element = this.first;
54          while (element != null && i > 0) {
55              element = element.getNext();
56              i--;
57          }
58          if (element == null) {
59              throw new IndexOutOfBoundsException();
60          }
61          return element.getPostkarte();
62      }
63  }
```

```
1   /**
2    * Diese Klasse repraesentiert ein Element in einer
3    * {@link MeinePostkartenListe}. Sie verweist auf die gespeicherte Postkarte
4    * fuer dieses Element und verweist ebenso auf das naechste Element vom Typ
5    * {@link MeinPostkartenListenElement} in der Liste.
6    *
7    * @author Annabelle Klarl
8    *
9    */
10  public class MeinPostkartenListenElement {
11
12      private Postkarte postkarte;
13      private MeinPostkartenListenElement next;
14
15      /**
16       * Konstruktor: erstellt ein neues Element mit einem Verweis auf die
17       * entsprechende Postkarte
18       *
19       * @param postkarte
20       */
21      public MeinPostkartenListenElement(Postkarte postkarte) {
22          this.postkarte = postkarte;
23          this.next = null;
24      }
25
26      /**
27       * Konstruktor: erstellt ein neues Element mit einem Verweis auf die
28       * entsprechende Postkarte und erstellt einen Verweis auf das naechste
29       * Listen-Element.
30       *
31       * @param postkarte
32       * @param next
33       */
34      public MeinPostkartenListenElement(
35              Postkarte postkarte, MeinPostkartenListenElement next) {
36          this.postkarte = postkarte;
37          this.next = next;
38      }
39
40      /**
41       * Diese Methode gibt die zu diesem Listen-Element gehoerende Postkarte
42       * zurueck.
43       *
44       * @return
45       */
```

```
46      public Postkarte getPostkarte() {
47          return this.postkarte;
48      }
49
50      /**
51       * Diese Methode gibt das naechste Listen-Element zurueck.
52       *
53       * @return
54       */
55      public MeinPostkartenListenElement getNext() {
56          return this.next;
57      }
58  }
```

c) Die grafische Benutzeroberfläche für die Verwaltung des Ballonwettbewerbs soll wie folgt aussehen:



Es soll zwei Buttons mit der oben genannten Aufschrift geben. Darunter soll ein Ausgabebereich platziert werden.

Schreiben Sie eine Klasse `MeinPostkartenFrame`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `MeinPostkartenFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `MeinPostkartenFrame` abspeichern.

Erweitern Sie Ihre Klasse `MeinPostkartenFrame` um eine Ereignisbehandlung für die beiden Buttons. Wird der Button "Neue Postkarte hinzufügen" gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach dem Namen des Teilnehmers und der zurückgelegten Entfernung gefragt werden. Anschließend soll aus diesen Angaben ein Objekt vom Typ `Postkarte` erstellt werden und in einer Liste vom Typ `MeinePostkartenListe` gespeichert werden. Darüber soll der Benutzer im Ausgabebereich informiert werden.

Wird der Button "Sieger aus allen bisherigen Postkarten berechnen" gedrückt, sollen in der Liste aller bisher eingegebenen Postkarten alle Postkarten gesucht werden, die die weiteste Entfernung erreicht haben. Der Benutzer soll anschließend darüber informiert werden, welche Postkarten die weiteste Entfernung erreicht haben. Beachten Sie, dass dies auch mehrere Postkarten sein können.

Hinweis: In Ihrer Klasse `MeinPostkartenFrame` werden Sie ein Attribut brauchen, um die Liste von Postkarten speichern zu können (Modell der GUI).

Lösung:
Bei der Implementierung der GUI geht man wie gewohnt vor. Da die Ausgabe der siegreichen Postkarten allerdings relativ lang sein kann, können Sie mit Hilfe der Klassen `ScrollPane`

und `ScrollPaneConstants` der Swing-Bibliothek dem Ausgabebereich auf folgende Weise eine vertikale Scrollbar hinzufügen.

```
JScrollPane scrollPane = new JScrollPane(this.ausgabeBereich);
scrollPane.setHorizontalScrollBarPolicy(
   ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
```

Vergessen Sie außerdem nicht Ihrer Klasse `MeinPostkartenFrame` ein Attribut vom Typ `MeinePostkartenListe` hinzuzufügen, das alle eingetragenen Postkarten speichern soll und bei der Benutzung des Buttons "Neue Postkarte hinzufügen" um ein Element erweitert wird.

Um alle Postkarten mit der größten Entfernung zu finden, geht man wie folgt vor:

1. Lege eine neue Liste zur Speicherung aller Sieger-Postkarten an.
2. Deklariere eine lokale Variable `groessteEntfernung`, die die maximale Entfernung einer Postkarte speichern soll, und initialisiere sie mit -1.
3. Durchlaufe die Liste von gespeicherten Postkarten und wiederhole für jedes Element:
   - Falls die Postkarte weiter geflogen ist als die aktuell größte Entfernung, lösche die bisherige Siegerliste. Füge die aktuelle Postkarte in die nun leere Siegerliste ein und speichere die Entfernung dieser Postkarte als nun aktuell größte Entfernung in der lokalen Variablen `groessteEntfernung`.
   - Falls die Postkarte genauso weit geflogen ist wie die aktuell größte Entfernung, füge die aktuelle Postkarte in die Siegerliste ein.
4. Gebe die Siegerliste aus.

Lösung:
Eine Implementierung der Klassen `MeinPostkartenFrameMain` und `MeinPostkartenFrame` finden Sie auch im ZIP-Archiv.

```
/**
 * Diese Klasse startet den MeinPostkartenFrame
 *
 * @author Annabelle Klarl
 */
public class MeinPostkartenFrameMain {

    /**
     * Dieses Programmstueck startet das Programm.
     *
     * @param args
     */
    public static void main(String[] args) {
        MeinPostkartenFrame frame = new MeinPostkartenFrame();
        frame.setVisible(true);
    }

}
```

```
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.ScrollPaneConstants;

```

```java
/**
 * Diese Klasse repraesentiert das Hauptfenster der Postkarten-Praesenzaufgabe,
 * wobei eine eigene verkettete Liste verwendet wird.
 *
 * @author Annabelle Klarl
 *
 */
public class MeinPostkartenFrame extends JFrame implements ActionListener {
    private JButton postkarteHinzufuegenButton;
    private JButton siegerBerechnenButton;

    private JTextArea ausgabeBereich;

    private MeinePostkartenListe postkartenListe;

    /**
     * In diesem Programmstueck wird das Fenster erzeugt
     */
    public MeinPostkartenFrame() {
        /* Verbindung zum Modell der GUI */
        this.postkartenListe = new MeinePostkartenListe();

        /* In der Kopfleiste des Fenster steht "MeinPostkartenFrame" */
        this.setTitle("MeinPostkartenFrame");

        /* Das Fenster ist 500 Pixel breit und 250 Pixel hoch. */
        this.setSize(500, 250);

        /* Hier werden alle Buttons erzeugt. */
        this.postkarteHinzufuegenButton = new JButton(
                "Neue Postkarte hinzufuegen");
        this.siegerBerechnenButton = new JButton(
                "Sieger aus allen bisherigen Postkarten berechnen");

        /* Hier wird der Ausgabe-Bereich erzeugt. */
        this.ausgabeBereich = new JTextArea(10, 100);
        JScrollPane scrollPane = new JScrollPane(this.ausgabeBereich);
        scrollPane.setHorizontalScrollBarPolicy(
                ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

        /*
         * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
         * Interaktionselemente (wie eine TextArea oder ein Button) platziert
         * werden koennen.
         */
        Container contentPane = this.getContentPane();
        contentPane.setLayout(new GridLayout(3, 1));
        /* Hier wird der Button platziert. */
        contentPane.add(this.postkarteHinzufuegenButton);
        contentPane.add(this.siegerBerechnenButton);
        /* Hier wird der Ausgabebereich platziert. */
        contentPane.add(scrollPane);

        /*
         * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei
         * jedem der Buttons registriert.
         */
        this.postkarteHinzufuegenButton.addActionListener(this);
        this.siegerBerechnenButton.addActionListener(this);

        /*
         * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
         * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
         */
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
78      }
79
80      @Override
81      public void actionPerformed(ActionEvent e) {
82          // je nach Button entsprechende Methode ausfuehren
83          Object source = e.getSource();
84          if (source == this.postkarteHinzufuegenButton) {
85              this.postkarteHinzufuegen();
86          } else if (source == this.siegerBerechnenButton) {
87              this.siegerBerechnen();
88          }
89      }
90
91      /**
92       * implementiert die Funktionalitaet des "Postkarte hinzufuegen"-Buttons
93       */
94      private void postkarteHinzufuegen() {
95          String name = JOptionPane.showInputDialog(
96                  "Name des Teilnehmers: ");
97
98          String einlesenEntfernung = JOptionPane.showInputDialog(
99                  "Zurueckgelegte Entfernung in km: ");
100         int entfernung = Integer.parseInt(einlesenEntfernung);
101
102         Postkarte neuePostkarte = new Postkarte(name, entfernung);
103         this.postkartenListe.addFirst(neuePostkarte);
104
105         this.ausgabeBereich.setText("Die Postkarte des Teilnehmers "
106                 + name + " mit einer Entfernung von " + entfernung
107                 + " km wurde hinzugefuegt.");
108     }
109
110     /**
111      * implementiert die Funktionalitaet des "Sieger berechnen"-Buttons
112      */
113     private void siegerBerechnen() {
114         int groessteEntfernung = -1;
115         MeinePostkartenListe siegerListe = new MeinePostkartenListe();
116
117         for (int i = 0; i < this.postkartenListe.size(); i++) {
118             Postkarte postkarte = this.postkartenListe.get(i);
119             int entfernung = postkarte.getEntfernung();
120
121             if (entfernung > groessteEntfernung) {
122                 groessteEntfernung = entfernung;
123                 siegerListe = new MeinePostkartenListe();
124                 siegerListe.addFirst(postkarte);
125             } else if (entfernung == groessteEntfernung) {
126                 siegerListe.addFirst(postkarte);
127             }
128             // else: diese Postkarte ist nicht weiter gereist
129             // als die bisherigen Siegerpostkarte
130         }
131
132         this.ausgabeBereich.setText("Die Sieger sind: \n");
133         for (int i = 0; i < siegerListe.size(); i++) {
134             Postkarte postkarte = siegerListe.get(i);
135             this.ausgabeBereich.append(postkarte.getTeilnehmer()
136                     + " mit einer Entfernung von "
137                     + postkarte.getEntfernung() + " km");
138             this.ausgabeBereich.append("\n");
139         }
140     }
141 }
```

d) Zur Lösung der Aufgabe sollen nun die generischen Klassen `LinkedList<E>` und `Iterator<E>` der Java-Bibliothek verwendet werden. Nehmen Sie die Benutzeroberfläche aus Aufgabe c) zur Grundlage und schreiben Sie eine neue Klasse `PostkartenFrame`. Das Aussehen der Benutzeroberfläche sowie die Funktionsweise der Buttons soll nicht verändert werden, allerdings soll anstelle eine Liste vom Typ `MeinePostkartenListe` nun eine Liste vom Typ `LinkedList<Postkarte>` verwendet werden, um alle Postkarten zu speichern. Ebenso soll zum Durchlaufen einer Liste stets ein Objekt der Klasse `Iterator<Postkarte>` benutzt werden. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `PostkartenFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `PostkartenFrame` abspeichern.

Lösung:
Eine Implementierung der Klassen `PostkartenFrameMain` und `PostkartenFrame` finden Sie auch im ZIP-Archiv.

```java
/**
 * Diese Klasse startet den PostkartenFrame
 *
 * @author Annabelle Klarl
 */
public class PostkartenFrameMain {

    /**
     * Dieses Programmstueck startet das Programm.
     *
     * @param args
     */
    public static void main(String[] args) {
        PostkartenFrame frame = new PostkartenFrame();
        frame.setVisible(true);
    }

}
```

```java
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Iterator;
import java.util.LinkedList;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.ScrollPaneConstants;

/**
 * Diese Klasse repraesentiert das Hauptfenster der Postkarten-Praesenzaufgabe,
 * wobei {@link LinkedList} verwendet wird.
 *
 * @author Annabelle Klarl
 *
 */
public class PostkartenFrame extends JFrame implements ActionListener {
    private JButton postkarteHinzufuegenButton;
    private JButton siegerBerechnenButton;

    private JTextArea ausgabeBereich;

    private LinkedList<Postkarte> postkartenListe;

```

```
30      /**
31       * In diesem Programmstueck wird das Fenster erzeugt
32       */
33      public PostkartenFrame() {
34          /* Verbindung zum Modell der GUI */
35          this.postkartenListe = new LinkedList<Postkarte>();
36
37          /* In der Kopfleiste des Fenster steht "PostkartenFrame" */
38          this.setTitle("PostkartenFrame");
39
40          /* Das Fenster ist 500 Pixel breit und 250 Pixel hoch. */
41          this.setSize(500, 250);
42
43          /* Hier werden alle Buttons erzeugt. */
44          this.postkarteHinzufuegenButton = new JButton(
45                  "Neue Postkarte hinzufuegen");
46          this.siegerBerechnenButton = new JButton(
47                  "Sieger aus allen bisherigen Postkarten berechnen");
48
49          /* Hier wird der Ausgabe-Bereich erzeugt. */
50          this.ausgabeBereich = new JTextArea(10, 100);
51          JScrollPane scrollPane = new JScrollPane(this.ausgabeBereich);
52          scrollPane.setHorizontalScrollBarPolicy(
53                  ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
54
55          /*
56           * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets
57           * d.h. Interaktionselemente (wie eine TextArea oder ein Button)
58           * platziert werden koennen.
59           */
60          Container contentPane = this.getContentPane();
61          contentPane.setLayout(new GridLayout(3, 1));
62          /* Hier wird der Button platziert. */
63          contentPane.add(this.postkarteHinzufuegenButton);
64          contentPane.add(this.siegerBerechnenButton);
65          /* Hier wird der Ausgabebereich platziert. */
66          contentPane.add(scrollPane);
67
68          /*
69           * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei
70           * jedem der Buttons registriert.
71           */
72          this.postkarteHinzufuegenButton.addActionListener(this);
73          this.siegerBerechnenButton.addActionListener(this);
74
75          /*
76           * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
77           * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
78           */
79          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
80      }
81
82      @Override
83      public void actionPerformed(ActionEvent e) {
84          // je nach Button entsprechende Methode ausfuehren
85          Object source = e.getSource();
86          if (source == this.postkarteHinzufuegenButton) {
87              this.postkarteHinzufuegen();
88          } else if (source == this.siegerBerechnenButton) {
89              this.siegerBerechnen();
90          }
91      }
92
93      /**
94       * implementiert die Funktionalitaet des "Postkarte hinzufuegen"-Buttons
```

```
95       */
96      private void postkarteHinzufuegen() {
97          String name = JOptionPane.showInputDialog(
98                  "Name des Teilnehmers: ");
99
100         String einlesenEntfernung = JOptionPane.showInputDialog(
101                 "Zurueckgelegte Entfernung in km: ");
102         int entfernung = Integer.parseInt(einlesenEntfernung);
103
104         Postkarte neuePostkarte = new Postkarte(name, entfernung);
105         this.postkartenListe.addFirst(neuePostkarte);
106
107         this.ausgabeBereich.setText("Die Postkarte des Teilnehmers "
108                 + name + " mit einer Entfernung von " + entfernung
109                 + " km wurde hinzugefuegt.");
110     }
111
112     /**
113      * implementiert die Funktionalitaet des "Sieger berechnen"-Buttons
114      */
115     private void siegerBerechnen() {
116         int groessteEntfernung = -1;
117         LinkedList<Postkarte> siegerListe = new LinkedList<Postkarte>();
118
119         Iterator<Postkarte> iteratorPostkarten = this.postkartenListe.iterator();
120         while (iteratorPostkarten.hasNext()) {
121             Postkarte postkarte = iteratorPostkarten.next();
122             int entfernung = postkarte.getEntfernung();
123
124             if (entfernung > groessteEntfernung) {
125                 groessteEntfernung = entfernung;
126                 siegerListe = new LinkedList<Postkarte>();
127                 siegerListe.addFirst(postkarte);
128             } else if (entfernung == groessteEntfernung) {
129                 siegerListe.addFirst(postkarte);
130             }
131             // else: diese Postkarte ist nicht weiter gereist
132             // als die bisherigen Siegerpostkarten
133         }
134
135         this.ausgabeBereich.setText("Die Sieger sind: \n");
136         Iterator<Postkarte> iteratorGewinner = siegerListe.iterator();
137         while (iteratorGewinner.hasNext()) {
138             Postkarte postkarte = iteratorGewinner.next();
139             this.ausgabeBereich.append(postkarte.getTeilnehmer()
140                     + " mit einer Entfernung von "
141                     + postkarte.getEntfernung() + " km");
142             this.ausgabeBereich.append("\n");
143         }
144     }
145 }
```

Erinnerung: Klausurvorbereitung 6 und 9 ECTS
Überlegen und formulieren Sie Fragen zu Themen aus der Vorlesung, bei denen Sie noch Probleme haben und senden Sie diese per Mail an Philipp Wendler (`philipp.wendler@lmu.de`). Die am meisten gestellten Fragen werden in der Zentralübung am 07.02.2018 behandelt.

Besprechung der Präsenzaufgaben in den Übungen am 02.02.2018 und 05.02.2018.